全国计算机技术与软件专业技术资格（水平）考试指定用书

软件评测师教程

柳纯录 主编　黄子河 陈渌萍 副主编
全国计算机技术与软件专业技术资格（水平）考试办公室组编



清华大学出版社



内容简介

本书作为全国计算机技术与软件专业技术资格（水平）考试指定用书，本书全面系统地涵盖了软件评测专业的知识。全书共20章，对软件评测的基本理论、软件评测技术、软件评测管理以及软件评测具体案例进行了系统的讲解，附录部分对常用的测试工具做了简要的介绍。

本书中涉及的一些实例全部取材于中国软件评测中心数十年来的精华，既对软件评测理论作出了良好的诠释，也是软件评测技术在实践中应用的具体体现。通过本书的学习，读者可以大幅度提高软件评测的实践能力。

本书既是软件评测师考试培训必备参考教材，也可供从事软件质量保证、开发、管理以及信息系统工程监理的技术人员使用。

图书在版编目（CIP）数据

软件评测师教程 / 柳纯录主编. —北京：清华大学出版社，2005.3
（全国计算机技术与软件专业技术资格（水平）考试指定用书）
ISBN 7-302-10536-7

I. 软… II. 柳… III. 软件–测试–工程技术人员–资格考核–教材 IV. TP311.5

中国版本图书馆CIP数据核字（2005）第012370号

出 版 者：清华大学出版社　　地　址：北京清华大学学研大厦
http://www.tup.com.cn　　邮　编：100084
社 总 机：010-62770175　　客户服务：010-62776969
组稿编辑：柴文强
文稿编辑：赵晓宁
印 刷 者：北京密云胶印厂
装 订 者：三河市金元装订厂
发 行 者：新华书店总店北京发行所
开　本：185×230　印张：44.75　字数：920千字
版　次：2005年3月第1版　2005年3月第2次印刷
书　号：ISBN 7-302-10536-7/TP・7151
印　数：8001～13000
定　价：60.00元

序

在国务院鼓励软件产业发展政策的带动下，我国软件业一年一大步，实现了跨越式发展，销售收入由2000年的593亿元增加到2003年的1633亿元，年均增长速度39.2%；2000年出口软件仅4亿美元，去年则达到20亿美元，三年中翻了两番多；全国"双软认证工作体系"已经规范运行，截止2003年11月底，认定软件企业8582家，登记软件产品18287个；11个国家级软件产业基地快速成长，相关政策措施正在落实；我国软件产业的国际竞争力日益提高。

在软件产业快速发展的带动下，人才需求日益迫切，队伍建设与时俱进，而作为规范软件专业人员技术资格的计算机软件考试已在我国实施了十余年，累计报考人数超过一百万，为推动我国软件产业的发展作出了重要贡献。

软件考试在全国率先执行了以考代评的政策，取得了良好的效果。为贯彻落实国务院颁布的《振兴软件产业行动纲要》和国家职业资格证书制度，国家人事部和信息产业部对计算机软件考试政策进行了重大改革：考试名称调整为计算机技术与软件专业技术资格（水平）考试；考试对象从狭义的计算机软件扩大到广义的计算机软件，涵盖了计算机技术与软件的各个主要领域（5个专业类别、3个级别层次和20个职业岗位资格）；资格考试和水平考试合并，采用水平考试的形式（与国际接轨，报考不限学历与资历条件），执行资格考试政策（各用人单位可以从考试合格者中择优聘任专业技术职务）；这是我国人事制度改革的一次新突破。此外，将资格考试政策延伸到高级资格，使考试制度更为完善。

信息技术发展快，更新快，要求从业人员不断适应和跟进技术的变化，有鉴于此，国家人事部和信息产业部规定对通过考试获得的资格（水平）证书实行每隔三年进行登记的制度，以鼓励和促进专业人员不断接受新知识、新技术、新法规的继续教育。考试设置的专业类别、职业岗位也将随着国民经济与社会发展而动态调整。

目前，我国计算机软件考试的部分级别已与日本信息处理工程师考试的相应级别实现了互认，以后还将继续扩大考试互认的级别和国家。

为规范培训和考试工作，信息产业部电子教育中心组织一批具有较高理论水平和丰富实践经验的专家编写了全国计算机技术与软件专业技术资格（水平）考试的教材和辅导用书，按照考试大纲的要求，全面介绍相关知识与技术，帮助考生学习和备考。



我们相信，经过全社会的共同努力，全国计算机技术与软件专业技术资格（水平）考试将会更加规范、科学，进而对培养信息技术人才，加快专业队伍建设，推动国民经济和社会信息化作出更大的贡献。

信息产业部副部长 娄勤俭

2004年6月

前　言

软件正进入测试时代——这是中国软件评测中心从近几年的亲身经历中获得的感受。

国家质检总局和国家信息产业部从 2001 年开始把软件产品质量列入年度抽查之列（国家质检总局的抽查简称为“国抽”，信息产业部的抽查简称为“行抽”）。

国家科技部从 2002 年开始，对“863”高科技项目中的软件相关课题采取“以测代评”的管理方式，请公正、权威的第三方评测机构进行三维 CAD、ERP、操作系统、网络计算机、国产数据库等专项测试，以测试结果作为项目是否通过验收的重要依据，也以测试结果的排序作为研发单位是否获得新的立项支持的重要依据。

我国几十年来一直沿用的专家鉴定会方式现在也开始有了改变——先对所开发的软件技术成果进行公正、权威的第三方测试，然后再提交给专家鉴定会；有的甚至索性以测试代替鉴定会。

国家工商行政管理总局、国家税务总局、最高人民检察院等部委在本行业应用信息系统选型过程中，在判别谁家研发的软件能够“入围”时，也采用由公正、权威的测试机构进行测试的手段。

最近几年刚刚兴起的信息工程监理也把测试，特别是软件测试作为重要手段——从过去的项目验收测试发展到信息系统监理；从过去的定性评价发展到让“数字”说话。

为适应软件进入测试时代的需要，促进软件产业发展，相当多的省、自治区、直辖市和计划单列市，例如上海、北京、辽宁、浙江、四川、重庆、安徽、青岛等，成立了本地区的软件评测中心。

相当多的大、中型软件企业强化了软件测试活动，不仅抓紧软件开发人员的自测环节，还配备了专职的软件测试人员和测试部门开展独立的专职测试。

于是，开始出现软件测试人员紧缺。人们开始强烈感到：软件进入测试时代，人才队伍建设是关键。为此一些高等院校审时度势，在多年来开设软件测试课程的基础上，开始筹建或已经设立了软件测试专业。

“软件评测师”被列入计算机技术与软件专业技术资格（水平）考试。2003 年 10 月 18 日，国家人事部和信息产业部联合发文（国人部发[2003]39 号），决定在多年来实行程序员、高级程序员（软件设计师）等专业技术资格（水平）考试取得成功经验的基础上，在“中级资格”中增加“软件评测师”一项。自此，软件评测人员开始有了独立的



专业技术资格名称，这不仅对于从事软件评测的人员是巨大鼓舞和鞭策，也表明了软件评测正在成为独立的行当，软件评测人员的地位和作用受到前所未有的重视，这既是软件进入测试时代的重要标志，也是推进软件测试行业发展的有力措施。

要造就数以万计的软件测试人才队伍，关键在于抓好教育培训；而一套先进实用的教材又成了教育培训的关键。中国软件评测中心凝聚 15 年专业测试成果和精华，吸取了国际和国内软件测试领域的经验，紧扣最新颁布的《软件评测师考试大纲》，参考了大量相关书籍和文献，在国内多位专家的指导和帮助下，编写了《软件评测师教程》一书。

本书内容较为全面地涵盖了软件评测专业的知识，追溯了软件测试的发展史，反映了当前国际上采用的最先进的测试理论、标准、技术和工具，展望了软件测试的发展趋势，强调了表述的准确性、知识的系统性以及技术的先进性和实用性。书中的软件测试术语、测试方法、测试标准统一到当前最新的软件工程、软件产品评价和产品质量等相关的国际、国家标准上。本书为了满足广大用户在信息系统建设中对测试提出的需求，特别注意到软件测试技术在信息系统测试中的应用。软件评测是理论与实践紧密结合的工作，为了使教程具有较强的实践性，本书提供了系统功能测试、白盒测试和性能故障定位与分析等示范案例，以便从事软件测试的人员能有更深刻的理解。同时本书还对前瞻性的软件测试技术和流行的测试工具作了一些介绍。

本书由柳纯录担任主编，黄子河、陈渌萍担任副主编。第 1 章由黄子河编写，第 2 章由高炽扬、罗文兵、黄民德编写，第 3、4 章由黄民德编写，第 5 章由黄江平编写，第 6 章由陶新昕编写，第 7 章由耿莉编写，第 8 章由陈渌萍编写，第 9 章由黄官银编写，第 10、11 章由袁志民、相春雷编写，第 12 章由朱璇编写，第 13 章由耿莉编写，第 14 章由杨巨森编写，第 15 章由黄官银编写，第 16 章由朱璇编写，第 17 章由董晓阳编写，第 18 章由陶新昕编写，第 19 章由罗文兵编写，第 20 章由陈渌萍编写，附录由郝煜编写，全书由柳纯录、黄子河、陈渌萍统稿。朱璇、黄官银、黄民德、相春雷也对全书的修改和完善做了大量的工作。

本书着重于考试大纲中的软件评测知识和软件测试技术与应用相关内容的深化和细化，有些与基础知识相关的内容则需参考相关的计算机技术与软件技术专业技术资格考试教程。

中国电子信息产业发展研究院、中国软件评测中心各位领导和同仁，信息产业部电子教育中心、全国计算机专业技术资格（水平）考试专家委员会的各位领导和专家以及国内其他多位专家，对本书的编写倾注了热情关怀、悉心指导和鼎力帮助，我们谨在此表示诚挚的感谢！

“软件评测师考试”是计算机技术与软件专业技术资格（水平）考试中新开设的门类，所以，编写《软件评测师教程》一书是一项具有强烈挑战的创新性工作，难度大，时间紧，加之编者水平有限，书中难免有疏漏之处，敬请广大读者不吝赐教。

如果有意见和建议请与中国软件评测中心联系。E-mail: cyh@cstc.org.cn。

编　者

2005年1月

目　　录

第一篇　理　论　篇



第二篇　测试技术





第三篇 测试案例



第一篇　理　论　篇

第1章　软件测试概论

1.1　概述

软件测试是伴随着软件的产生而产生的，有了软件生产和运行就必然有软件测试。早期的软件开发过程中，测试的含义比较狭窄，将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对测试的投入极少，测试介入得也晚，常常是等到形成代码，产品已经基本完成时才进行测试。

直到 1957 年，软件测试才开始与调试区别开来，成为一种发现软件缺陷的活动。由于一直存在着为了使我们看到产品在工作，就得将测试工作往后推一点的思想，测试仍然是后于开发的活动。在潜意识里，我们的目的是使自己确信产品能工作。到了 20 世纪 70 年代，尽管对“软件工程”的真正含义还缺乏共识，但这一词条已经频繁出现。1972 年在北卡罗来纳大学举行了首届软件测试正式会议，1975 年 John Good Enough 和 Susan Gerhart 在 IEEE 上发表了“测试数据选择的原理 （Toward a Theory of Test Data Selection)”的文章，软件测试才被确定为一种研究方向。而 1979 年，Glen ford Myers 的《软件测试艺术》(The Art of Software Testing）可算是软件测试领域的第一本最重要的专著，Myers 作为当时最好的软件测试，其定义是：“测试是为发现错误而执行的一个程序或者系统的过程”。Myers 以及他的同事们在 20 世纪 70 年代的工作是测试过程发展的里程碑。

直到 20 世纪 80 年代早期，“质量“的号角才开始吹响。软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且包含软件质量评价的内容。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。制定了各类标准，包括 IEEE（Institute of Electrical and Electronic Engineers）标准、美国 ANSI（American National Standard Institute）标准以及 ISO（International Standard Organization）国际标准。1983 年，Bill Hetzel 在《软件测试完全指南》(Complete Guide of Software Testing）一书中指出：“测试是以评价一个程序或者系统属性为目标的任何一种活动，测试是对软件质量的度量”。Myers 和 Hetzel 的定义至今仍被引用。

20 世纪 90 年代，测试工具终于盛行起来。人们普遍意识到工具不仅是有用的，而且要对今天的软件系统进行充分的测试，工具是必不可少的。到了 2002 年，Rick 和 Stefan 在《系统的软件测试》（Systematic Software Testing）一书中对软件测试做了进一步定义："测试是为了度量和提高被测软件的质量，对测试软件进行工程设计、实施和维护的整个生命周期过程"。这些经典论著对软件测试研究的理论化和体系化产生了巨大的影响。

近 20 年来，随着计算机和软件技术的飞速发展，软件测试技术研究也取得了很大的突破，测试专家总结了很好的测试模型，比如著名的 V 模型、W 模型等，在测试过程改进方面提出了 TMM（Testing Maturity Model）的概念，在单元测试、自动化测试、负载压力测试以及测试管理等方面涌现了大量优秀的软件测试工具。

虽然软件测试技术的发展很快，但是其发展速度仍落后于软件开发技术的发展速度，使得软件测试在今天面临着很大的挑战，主要体现在以下几个方面。

① 软件在国防现代化、社会信息化和国民经济信息化领域中的作用越来越重要，由此产生的测试任务越来越繁重。

② 软件规模越来越大，功能越来越复杂，如何进行充分而有效的测试成为难题。

③ 面向对象的开发技术越来越普及，但是面向对象的测试技术却刚刚起步。

④ 对于分布式系统整体性能还不能进行很好的测试。

⑤ 对于实时系统来说，缺乏有效的测试手段。

⑥ 随着安全问题的日益突出，信息系统的安全性如何进行有效的测试与评估，成为世界性的难题。

1.2 国内外现状

在软件比较发达的国家，特别是美国，软件测试已经发展成为一个独立的产业，主要体现在以下几个方面。

① 软件测试在软件公司中占有重要的地位。比尔·盖茨曾在马萨诸塞州技术学院的一次演讲中说："在微软，一个典型的开发项目组中测试工程师要比编码工程师多得多，可以说我们花费在测试上的时间要比花费在编码上的时间多得多"。

② 软件测试理论研究蓬勃发展，每年举办各种各样的测试技术年会，发表了大量的软件测试研究论文，引领软件测试理论研究的国际潮流。

③ 软件测试市场繁荣。美国有一些专业公司开发软件测试标准与测试工具，MI、Compuware、MaCabe、Rational 等都是著名的软件测试工具提供商，它们出品的测试工具已经占领了国际市场，目前我国使用的主流测试工具大部分是国外的产品，而且在世界各地都可以看到它们出品的软件测试工具，可见国外的软件测试已经形成了较大的

产业。

中国的软件测试技术研究起步于“六五”期间，主要是随着软件工程的研究而逐步发展起来的，由于起步较晚，与国际先进水平相比差距较大。直到 1990 年，成立了国家级的中国软件评测中心，测试服务才逐步开展起来。因此，我国无论是在软件测试理论研究还是在测试实践上，和国外发达国家都有不少的差距，主要体现在对软件产品化测试的技术研究还比较贫乏，从业人员较少，测试服务没有形成足够的规模等方面。但是，随着我国软件产业的蓬勃发展以及对软件质量的重视，软件测试也越来越被人们所看重，软件测试正在逐步成为一个新兴的产业。我国正在迈入测试时代，主要体现在以下几个方面。

① 我国著名的软件公司都已经或者正在建立独立的专职软件测试队伍，虽然测试人员规模以及所占比例还不能和国外的大公司相比，但是毕竟在公司内部贯彻了独立测试的意识。

② 国家人事部和信息产业部 2003 年关于职业资格认证第一次在我国有了“软件评测师”的称号，这是国家对软件测试职业的高度重视与认可。

③ 在信息产业部关于计算机系统集成资质以及信息系统工程监理资质的认证中，软件测试能力已经被定为评价公司技术能力的一项重要指标。

④ 2001 年信息产业部发布的部长 5 号令，实行了软件产品登记认证制度，规定，凡是在我国境内销售的产品必须到信息产业部备案登记，而且要经过登记测试。

⑤ 自 2001 年起，国家质检总局和信息产业部每年都通过测试对软件产品进行质量监督抽查。

⑥ 国家各部委，各行业正在通过测试规范行业软件的健康发展，通过测试把不符合行业标准要求的软件挡在了门外，对行业信息化的健康发展起到了很好的促进作用。

⑦ 用户对软件质量要求越来越高，信息系统验收不再走过场，而要通过第三方测试机构的严格测试来判定。

⑧ “以测代评”正在成为我国科技项目择优支持的一项重要举措，比如，国家“863”计划对数据库管理系统、操作系统、办公软件、ERP 等项目的经费支持，都是通过第三方测试机构科学客观的测试结果来决定的。

⑨ 软件测试正在成为部分软件学院的一门独立课程，对我国软件测试人才的培养起到了很好的作用。

⑩ 第三方测试机构得到了蓬勃的发展。最近两年，在全国各地，新成立的软件测试机构有 10 多家，测试服务体系已经基本确立。

可见我国的软件测试行业正处于一个快速成长的阶段，我们有理由相信，经过一段时间的发展，我们会逐步缩小与国外发达国家的差距，从而带动整个软件产业的健康

发展。

1.3 软件测试与软件项目的关系

软件测试是为软件项目服务的，在整个项目组中，要强调测试服务的概念，虽然软件测试的目的是为了发现软件中存在的错误，但是，其根本目的是为了提高软件质量，降低软件项目的风险。软件的质量风险表现在两个方面，一种是内部风险，一种是外部风险。内部风险是在即将销售的时候发现有重大的错误，从而延迟发布日期，失去市场机会；外部风险是用户发现了不能容忍的错误，引起索赔、法律纠纷，以及用于客户支持的费用甚至失去客户的风险。

软件测试只能证明软件存在错误，而不能证明软件没有错误。软件公司对软件项目的期望是在预计的时间、合理的预算下，提交一个可以交付的产品，测试的目的就是把软件的错误控制在一个可以进行产品交付/发布的程度上，可以交付/发布的产品并不是没有错误的产品，因此软件测试不可能无休止地进行下去，而是要把错误控制在一个合理的范围之内，因为软件测试也是需要花费巨大成本的。有资料表明，波音 777 整体设计费用的 25%都花在了软件的 MC/DC（修正条件判定覆盖测试，是单元白盒测试的一种方法）测试上了，而且随着测试时间的延伸，发现错误的成本会越来越大，这就需要测试有度，而这个度并不能由项目计划时间来判断，而是要根据测试出现错误的概率来判断。这也要求在项目计划时，要给测试留出足够的时间和经费，仓促的测试或者由于项目提交计划的压力而终止测试，只能对整个项目造成无法估量的损害。

1.4 软件测试的发展趋势

纵观国内外软件测试的发展现状，可以看到软件测试有以下的发展趋势。

① 测试工作将进一步前移。软件测试不仅仅是单元测试、集成测试、系统测试和验收测试，对需求的精确性和完整性的测试技术、对系统设计的测试技术将成为新的研究热点。

② 软件架构师、开发工程师、QA 人员、测试工程师将进行更好的融合。他们相互之间要成为伙伴关系，而不是相互对立的关系，因为他们的工作可以相互借鉴，相互促进，而且软件测试工程师应该尽早地介入整个工程，在软件定义阶段就要开发相应的测试方法，使得每一个需求定义都是可以测试的。

③ 测试职业将得到充分的尊重。测试工程师和开发工程师不仅是矛盾体，也是相互协调的统一体。在整个软件开发周期，他们提供的是一种至关重要的服务，人们将充

分认识到测试的价值。以前人们认为“如果你没有能力做开发，那么就去做测试”，而现在的流行观点是“只有高水平的开发者，才能胜任测试工作”。

④ 设置独立的软件测试部门将成为越来越多的软件公司的共识。软件测试部门将和开发部、质量保证部一样作为一个重要的独立部门存在。

⑤ 测试外包服务将快速增长。和软件开发外包一样，软件测试外包将成为全球化的一种趋势，可以利用职业测试专家队伍与机构为自己的产品进行测试，而且可以节省测试费用。

1.5　第三方测试

这里所说的第三方测试是指独立于软件公司自身测试的测试。所谓的第三方是指在软件公司和软件用户之间的一方。第三方测试机构也是一个中介的服务机构，它通过自己专业化的测试手段为客户提供有价值的服务。但是第三方测试机构提供的服务不同于公司内部的测试。因为，第三方测试机构的测试除了发现软件问题之外，还有对软件进行科学、公正的评价的职能，这就要求第三方测试机构要保持公正、廉洁、客观、科学、独立的态度。

第三方测试机构存在的价值主要是由软件公司、软件用户以及国家的公正诉求所决定的。对于软件开发商来说，经过第三方测试机构的测试，不仅可以通过专业化的测试手段发现软件错误，帮助开发商提升软件的品质，而且可以对软件有一个客观、科学的评价，有助于开发商认清自己产品的定位。对于行业主管部门以及软件使用者来说，第三方测试机构独立公正的地位有助于对被测软件进行客观公正的评价，帮助用户选择合适、优秀的软件产品。而对于一些信息工程项目来说，在验收之前，经过第三方机构的严格测试，可以最大程度地避免信息行业的“豆腐渣”工程。此外，经过国家认可的第三方测试机构，还为国家软件产品的质量监督抽查提供独立公正的测试支持。

由此可见，第三方测试机构的测试工程师面对的是各种各样的系统，而且大多与具体的业务相关，这就要求他们不仅有宽广深厚的软件技术功底、测试技术功底，而且需要积累行业知识和经验，并且要融会贯通。目前，我国涌现了很多的第三方测试机构，虽然它们处于不同的发展阶段，但是它们的存在必将对我国整个软件产业的健康发展起到巨大的促进作用。

第 2 章　软件测试基础

2.1　软件测试与软件质量

2.1.1　什么是软件测试

测试（test）最早出于古拉丁字，它有“罐”或“容器”的含义。在工业制造和生产中，测试被当作一个常规的检验产品质量的生产活动。测试的含义为“以检验产品是否满足需求为目标”。而软件测试活动包括了很重要的任务，即发现错误。

“软件测试”的经典定义是在规定条件下对程序进行操作，以发现错误，对软件质量进行评估。

我们知道，软件是由文档、数据以及程序组成的，那么软件测试就应该是对软件形成过程的文档、数据以及程序进行的测试，而不仅仅是对程序进行的测试。

随着人们对软件工程化的重视以及软件规模的日益扩大，软件分析、设计的作用越来越突出，而且有资料表明，60%以上的软件错误并不是程序错误，而是分析和设计错误。因此，做好软件需求和设计阶段的测试工作就显得非常重要。这就是我们提倡的测试概念扩大化，提倡软件全生命周期测试的理念。

2.1.2　什么是软件质量

在 1991 年软件产品质量评价国际标准 ISO 9126 中定义的“软件质量”是：软件满足规定或潜在用户需求特性的总和。到 1999 年，软件“产品评价”国际标准 ISO 14598 经典的“软件质量”定义是：软件特性的总和，软件满足规定或潜在用户需求的能力。

一般对“质量”的理解是一个实体的“属性”，“属性”好就是质量好的。但是这不够全面，“属性”是内在特性，内在特性好，不一定能胜任和完成好用户的任务。因此，软件质量也是关于软件特性具备“能力”的体现。

2001 年，软件“产品质量”国际标准 ISO 9126 定义的软件质量包括“内部质量”、“外部质量”和“使用质量”三部分。也就是说，“软件满足规定或潜在用户需求的能力”要从软件在内部、外部和使用中的表现来衡量。

2.1.3 软件测试与质量保证的区别

软件测试人员的一项重要任务是提高软件质量，但不等于说软件测试人员就是软件质量保证人员，因为测试只是质量保证工作中的一个环节。软件质量保证和软件测试是软件质量工程的两个不同层面的工作。

- 质量保证（QA）：质量保证的重要工作通过预防、检查与改进来保证软件质量。QA 采用“全面质量管理”和“过程改进”的原理开展质量保证工作。所关注的是软件质量的检查与测量。虽然在 QA 的活动中也有一些测试活动，但所关注的是软件质量的检查与测量。QA 的工作是软件生命周期的管理以及验证软件是否满足规定的质量和用户的需求，因此主要着眼于软件开发活动中的过程、步骤和产物，而不是对软件进行剖析找出问题或评估。
- 软件测试：测试虽然也与开发过程紧密相关，但关心的不是过程的活动，而是对过程的产物以及开发出的软件进行剖析。测试人员要“执行”软件，对过程中的产物——开发文档和源代码进行走查，运行软件，以找出问题，报告质量。测试人员必须假设软件存在潜在的问题，测试中所作的操作是为了找出更多的问题，而不仅仅是为了验证每一件事是正确的。对测试中发现的问题的分析、追踪与回归测试也是软件测试中的重要工作，因此软件测试是保证软件质量的一个重要环节。

2.2 软件测试目的

早期的软件定义指出软件测试的目的是寻找错误，并且尽最大的可能找出最多的错误。

Grenford J. Myers 就软件测试目的提出了以下观点。

- 测试是程序的执行过程，目的在于发现错误；
- 一个好的测试用例在于能发现至今未发现的错误；
- 一个成功的测试是发现了至今未发现的错误的测试。

Bill Hetzel 提出了测试目的不仅仅是为了发现软件缺陷与错误，而且也是对软件质量进行度量和评估，以提高软件的质量。

测试的目的，是想以最少的人力、物力和时间找出软件中潜在的各种错误和缺陷，通过修正各种错误和缺陷提高软件质量，回避软件发布后由于潜在的软件缺陷和错误造成的隐患所带来的商业风险。

同时，测试是以评价一个程序或者系统属性为目标的活动，测试是对软件质量的度



量与评估，以验证软件的质量满足用户的需求的程度，为用户选择与接受软件提供有力的依据。

此外，通过分析错误产生的原因还可以帮助发现当前开发工作所采用的软件过程的缺陷，以便进行软件过程改进。同时，通过对测试结果的分析整理，还可以修正软件开发规则，并为软件可靠性分析提供依据。

当然，通过最终的验收测试，也可以证明软件满足了用户的需求，树立人们使用软件的信心。

2.3 软件测试原则

基于测试是为了寻找软件的错误与缺陷，评估与提高软件质量，我们提出这样的一组测试原则，如下所示。

- 所有的软件测试都应追溯到用户需求。

这是因为软件的目的是使用户完成预定的任务，并满足用户的需求，而软件测试所揭示的缺陷和错误使软件达不到用户的目标，满足不了用户需求。

- 应当把“尽早地和不断地进行软件测试”作为软件测试者的座右铭。

由于软件的复杂性和抽象性，在软件生命周期各个阶段都可能产生错误，所以不应把软件测试仅仅看作是软件开发的一个独立阶段的工作，而应当把它贯穿到软件开发的各个阶段中。在软件开发的需求分析和设计阶段就应开始测试工作，编写相应的测试文档。同时，坚持在软件开发的各个阶段进行技术评审与验证，这样才能在开发过程中尽早发现和预防错误，杜绝某些缺陷和隐患，提高软件质量。只要测试在生命周期中进行得足够早，就能够提高被测软件的质量，这就是预防性测试的基本原则。

- 完全测试是不可能的，测试需要终止。

想要进行完全的测试，在有限的时间和资源条件下，找出所有的软件缺陷和错误，使软件趋于完美，是不可能的。主要有三个原因：

① 输入量太大；

② 输出结果太多；

③ 路径组合太多。

一个适度规模的程序，其路径组合近似天文数字，对于每一种可能的路径都执行一次的穷举测试是不可能的。此外，测试也是有成本的，越是测试后期，为发现错误所付出的代价就会越大，因此也要根据测试错误的概率以及软件可靠性要求，确定最佳停止测试时间，我们不能无限地测试下去。

- 测试无法显示软件潜在的缺陷。

进行测试是可以查找并报告发现的软件缺陷和错误，但不能保证软件的缺陷和错误全部找到，继续进一步测试可能还会找到一些，也就是说测试只能证明软件存在错误而不能证明软件没有错误。

- 充分注意测试中的群集现象。

经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目或检错率成正比。根据这个规律，应当对错误群集的程序段进行重点测试，以提高测试投资的效益。

在所测程序段中，若发现错误数目多，则残存错误数目也比较多。这种错误群集性现象，已为许多程序的测试实践所证实。例如，在美国 IBM 公司的 OS/370 操作系统中，47%的错误仅与该系统的 4%的程序模块有关。这种现象对测试很有用。如果发现某一程序模块似乎比其他程序模块有更多的错误倾向，则应当花费较多的时间和代价测试这个程序模块。

- 程序员应避免检查自己的程序。

基于心理因素，人们认为揭露自己程序中的问题总不是一件愉快的事，不愿否认自己的工作；由于思维定势，人们难于发现自己的错误。因此，为达到测试目的，应由客观、公正、严格的独立的测试部门或者独立的第三方测试机构进行测试。

- 尽量避免测试的随意性。

应该从工程的角度去理解软件测试，它是有组织、有计划、有步骤的活动。

2.4　软件测试对象

根据软件定义，软件包括程序、数据和文档，所以软件测试并不仅仅是程序测试。软件测试应贯穿于整个软件生命周期中。在整个软件生命周期中，各阶段有不同的测试对象，形成了不同开发阶段的不同类型的测试。需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计规格说明、详细设计规格说明以及源程序，都应成为“软件测试”的对象。在软件编码结束后，对编写的每一个程序模块进行测试，称为“模块测试”或“单元测试”；在模块集成后，对集成在一起的模块组件，有时也可称为“部件”，进行测试，称为“集成测试”；在集成测试后，需要检测与证实软件是否满足软件需求说明书中规定的要求，这就称为“确认测试”。将整个程序模块集成为软件系统，安装在运行环境下，对硬件、网络、操作系统及支撑平台构成的整体系统进行测试，称为“系统测试”。

由于软件分析、设计与开发各阶段是互相衔接的，前一阶段工作中发生的问题如未及时解决，很自然要影响到下一阶段。从源程序的测试中找到的程序错误不一定都是在程序编写过程中产生的。如果简单地把程序中的错误全都归罪于程序员，未免冤枉了他



们。据美国一家公司的统计表明，在查找出的软件错误中，属于需求分析和软件设计的错误约占 64%，属于程序编写的错误仅占 36%。这都说明，对程序编写而言，它的许多错误是“先天的”。事实上，到程序的测试为止，软件开发工作已经经历了许多环节，每个环节都可能发生问题。

为了把握各个环节的正确性，人们需要进行各种验证和确认（verification & validation）工作。

验证（verification）是保证软件正确实现特定功能的一系列活动和过程，目的是保证软件生命周期中的每一个阶段的成果满足上一个阶段所设定的目标。

确认（validation）是保证软件满足用户需求的一系列的活动和过程，目的是在软件开发完成后保证软件与用户需求相符合。

验证与确认都属于软件测试，它包括对软件分析、设计以及程序的验证和确认。

2.5 软件测试分类

按照全生命周期的软件测试概念，测试对象应该包括软件设计开发的各个阶段的内容，对于需求和设计阶段的测试以及关于文档的测试将在面向对象与文档测试部分进行描述，这里重点讲述开发阶段的测试和程序测试。

2.5.1 按照开发阶段划分

按照开发阶段划分软件测试可分为：单元测试、集成测试、系统测试、确认测试和验收测试。

- 单元测试。

单元测试又称模块测试，是针对软件设计的最小单位——程序模块进行正确性检验的测试工作。其目的在于检查每个程序单元能否正确实现详细设计说明中的模块功能、性能、接口和设计约束等要求，发现各模块内部可能存在的各种错误。单元测试需要从程序的内部结构出发设计测试用例。多个模块可以平行地独立进行单元测试。

- 集成测试。

集成测试也叫做组装测试。通常在单元测试的基础上，将所有的程序模块进行有序的、递增的测试。集成测试是检验程序单元或部件的接口关系，逐步集成为符合概要设计要求的程序部件或整个系统。

软件集成的过程是一个持续的过程，会形成很多个临时版本，在不断的集成过程中，功能集成的稳定性是真正的挑战。在每个版本提交时，都需要进行冒烟测试，即对程序主要功能进行验证。冒烟测试也叫版本验证测试、提交测试。

- 确认测试。

确认测试是通过检验和提供客观证据，证实软件是否满足特定预期用途的需求。确认测试是检测与证实软件是否满足软件需求说明书中规定的要求。

- 系统测试。

系统测试是为验证和确认系统是否达到其原始目标，而对集成的硬件和软件系统进行的测试。系统测试是在真实或模拟系统运行的环境下，检查完整的程序系统能否和系统（包括硬件、外设、网络和系统软件、支持平台等）正确配置、连接，并满足用户需求。

- 验收测试。

按照项目任务书或合同、供需双方约定的验收依据文档进行的对整个系统的测试与评审，决定是否接收或拒收系统。

2.5.2 按照测试实施组织划分

按照测试实施组织划分，软件测试可分为开发方测试、用户测试（β 测试）、第三方测试。

- 开发方测试。

通常也叫“验证测试”或“α 测试”。开发方通过检测和提供客观证据，证实软件的实现是否满足规定的需求。验证测试是在软件开发环境下，由开发者检测与证实软件的实现是否满足软件设计说明或软件需求说明的要求。主要是指在软件开发完成以后，开发方对要提交的软件进行全面的自我检查与验证，可以和软件的“系统测试”一并进行。

- 用户测试。

在用户的应用环境下，用户通过运行和使用软件，检测与核实软件实现是否符合自己预期的要求。通常情况用户测试不是指用户的“验收测试”，而是指用户的使用性测试，由用户找出软件的应用过程中发现的软件的缺陷与问题，并对使用质量进行评价。

β 测试通常被看成是一种“用户测试”。β 测试主要是把软件产品有计划地免费分发到目标市场，让用户大量使用，并评价、检查软件。通过用户各种方式的大量使用，来发现软件存在的问题与错误，把信息反馈给开发者修改。β 测试中厂商获取的信息，可以有助于软件产品的成功发布。

- 第三方测试。

介于软件开发方和用户方之间的测试组织的测试。第三方测试也称为独立测试。软件质量工程强调开展独立验证和确认（IV&V）活动。IV&V 是由在技术、管理和财务上与开发组织具有规定程度独立的组织执行验证和确认过程。软件第三方测试也就是由在技术、管理和财务上与开发方和用户方相对独立的组织进行的软件测试。一般情况下是在模拟用户真实应用环境下，进行软件确认测试。



2.5.3 按照测试技术划分

按照测试技术划分：白盒测试、黑盒测试、灰盒测试。也可划分为静态测试和动态测试。静态测试是指不运行程序，通过人工对程序和文档进行分析与检查；静态测试技术又称为静态分析技术，静态测试实际上是对软件中的需求说明书、设计说明书、程序源代码等进行非运行的检查，静态测试包括：走查、符号执行、需求确认等。动态测试是指通过人工或使用工具运行程序进行检查、分析程序的执行状态和程序的外部表现。我们这里讨论的白盒测试、黑盒测试、灰盒测试，在实现测试方法上既包括了动态测试也包括了静态测试。

- 白盒测试

通过对程序内部结构的分析、检测来寻找问题。白盒测试可以把程序看成装在一个透明的白盒子里，也就是清楚了解程序结构和处理过程，检查是否所有的结构及路径都是正确的，检查软件内部动作是否按照设计说明的规定正常进行。白盒测试又称结构测试。

- 黑盒测试

通过软件的外部表现来发现其缺陷和错误。黑盒测试法把测试对象看成一个黑盒子，完全不考虑程序内部结构和处理过程。黑盒测试是在程序界面处进行测试，它只是检查样序是否按照需求规格说明书的规定正常实现。

- 灰盒测试

介于白盒测试与黑盒测试之间的测试。灰盒测试关注输出对于输入的正确性；同时也关注内部表现，但这种关注不像白盒测试那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态。

灰盒测试结合了白盒测试和黑盒测试的要素。它考虑了用户端、特定的系统知识和操作环境。它在系统组件的协同性环境中评价应用软件的设计。

软件测试方法和技术的分类与软件开发过程相关联，它贯穿了整个软件生命周期。走查、单元测试、集成测试、系统测试用于整个开发过程中的不同阶段。开发文档和源程序可以应用单元测试应用走查的方法；单元测试可应用白盒测试方法；集成测试应用近似灰盒测试方法；而系统测试和确认测试应用黑盒测试方法。

2.6 软件测试过程模型

在软件开发几十年的实践过程中，人们总结了很多的开发模型，比如瀑布模型、原型模型、螺旋模型、增量模型、渐进模型、快速软件开发（RAD）以及最近比较流行的

Rational 统一过程（RUP）等，这些模型对于软件开发过程具有很好的指导作用，但是，非常遗憾的是，在这些过程方法中，并没有充分强调测试的价值，也没有给测试以足够的重视，利用这些模型无法更好地指导测试实践。软件测试是与软件开发紧密相关的一系列有计划的系统性的活动，显然软件测试也需要测试模型去指导实践，非常可喜的是软件测试专家通过测试实践总结出了很多很好的测试模型。当然由于测试与开发的结合非常紧密，在这些测试模型中也都把开发过程进行了很好的总结，体现了测试与开发的融合，下面对主要的模型做一简单的介绍。

2.6.1　V 模型

V 模型是最具有代表意义的测试模型，如图 2-1 所示。V 模型最早是由 Paul Rook 在 20 世纪 80 年代后期提出的，V 模型在英国国家计算中心文献中发布，旨在改进软件开发的效率和效果。

在传统的开发模型中，比如瀑布模型，人们通常把测试过程作为在需求分析、概要设计、详细设计和编码全部完成之后的一个阶段，尽管有时测试工作会占用整个项目周期一半的时间，但是有人仍然认为测试只是一个收尾工作，而不是主要的过程。V 模型的推出就是对此种认识的改进。V 模型是软件开发瀑布模型的变种，它反映了测试活动与分析和设计的关系，从左到右，描述了基本的开发过程和测试行为，非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程期间各阶段的对应关系，如模型图（图 2-1）中所示，图中的箭头代表了时间方向，左边下降的是开发过程各阶段，与此相对应的是右边上升的部分，即各测试过程的各个阶段。

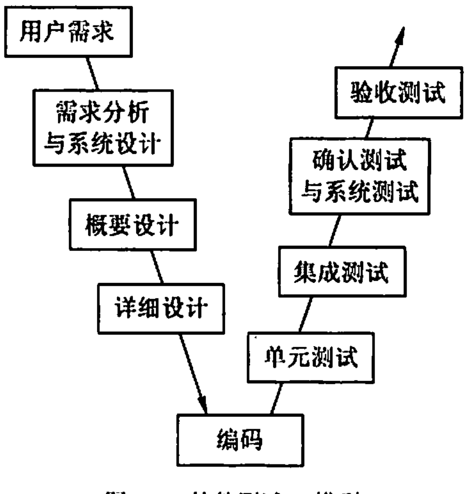


图 2-1　软件测试 V 模型



V 模型的软件测试策略既包括低层测试又包括了高层测试，低层测试是为了源代码的正确性，高层测试是为了使整个系统满足用户的需求。

V 模型指出，单元和集成测试是验证的程序设计，开发人员和测试组应检测程序的执行是否满足软件设计的要求；系统测试应当验证系统设计，检测系统功能、性能的质量特性是否达到系统设计的指标；由测试人员和用户进行软件的确认测试和验收测试，追溯软件需求说明书进行测试，以确定软件的实现是否满足用户需求或合同的要求。

V 模型存在一定的局限性，它仅仅把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段。容易使人理解为测试是软件开发的最后的一个阶段，主要是针对程序进行测试寻找错误，而需求分析阶段隐藏的问题一直到后期的验收测试才被发现。

2.6.2 W 模型

1. W 模型建立

V 模型的局限性在于没有明确地说明早期的测试，不能体现“尽早地和不断地进行软件测试”的原则。在 V 模型中增加软件各开发阶段应同步进行的测试，被演化为一种 W 模型，因为实际上开发是“V”，测试也是与此相并行的“V”。基于“尽早地和不断地进行软件测试”的原则，在软件的需求和设计阶段的测试活动应遵循 IEEE std 1012-1998 《软件验证和确认（V&V）》的原则。

一个基于 V&V 原理的 W 模型示意图如图 2-2 所示。

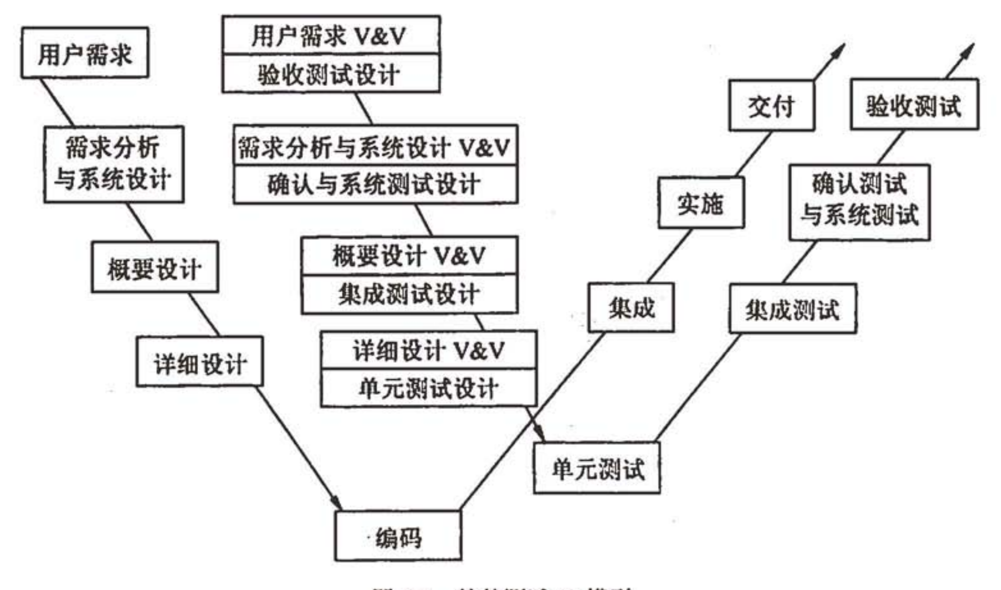


图 2-2 软件测试 W 模型

2. W 模型应用

W 模型由 Evolutif 公司提出，相对于 V 模型，W 模型更科学。W 模型可以说是 V 模型自然而然的发展。它强调：测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、功能和设计同样要测试。这样，只要相应的开发活动完成，我们就可以开始执行测试，可以说，测试与开发是同步进行的，从而有利于尽早地发现问题。以需求为例，需求分析一完成，我们就可以对需求进行测试，而不是等到最后才进行针对需求的验收测试。

如果测试文档能尽早提交，那么就有了更多的检查和检阅的时间，这些文档还可用于评估开发文档。另外还有一个很大的益处是，测试者可以在项目中尽可能早地面对规格说明书的挑战。这意味着测试不仅仅是评定软件的质量，测试还可以尽可能早地找出缺陷所在，从而帮助改进项目内部的质量。参与前期工作的测试者可以预先估计问题和难度，这将可以显著地减少总体测试时间，加快项目进度。

根据 W 模型的要求，一旦有文档提供，就要及时确定测试条件，以及编写测试用例，这些工作对测试的各级别都有意义。当需求被提交后，就需要确定高级别的测试用例来测试这些需求。当概要设计编写完成后，就需要确定测试条件来查找该阶段的设计缺陷。

W 模型也是有局限性的。W 模型和 V 模型都把软件的开发视为需求、设计、编码等一系列串行的活动。同样的，软件开发和测试保持一种线性的前后关系，需要有严格的指令表示上一阶段完全结束，才可正式开始下一个阶段。这样就无法支持迭代、自发性以及变更调整。对于当前很多文档需要事后补充，或者根本没有文档的做法下（这已成为一种开发的文化），开发人员和测试人员都面临同样的困惑。

2.6.3 H 模型

1. H 模型建立

V 模型和 W 模型均存在一些不妥之处。首先，如前所述，它们都把软件的开发视为需求、设计、编码等一系列串行的活动，而事实上，虽然这些活动之间存在互相牵制的关系，但在大部分时间内，它们是可以交叉进行的。虽然软件开发期望有清晰的需求、设计和编码阶段，但实践告诉我们，严格的阶段划分只是一种理想状况。试问，有几个软件项目是在有了明确的需求之后才开始设计的呢？所以，相应的测试之间也不存在严格的次序关系。同时，各层次之间的测试也存在反复触发、迭代和增量关系。其次，V 模型和 W 模型都没有很好地体现测试流程的完整性。

为了解决以上问题，有专家提出了 H 模型。它将测试活动完全独立出来，形成一个完全独立的流程，将测试准备活动和测试执行活动清晰地体现出来。



2．H 模型应用

H 模型的简单示意图如图 2-3 所示。

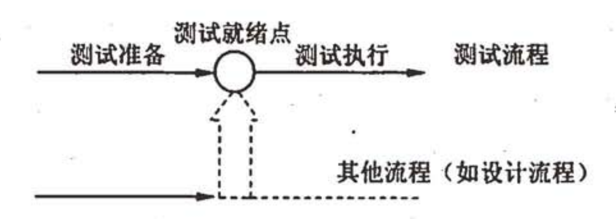


图 2-3　软件测试 H 模型

这个示意图仅仅演示了在整个生产周期中某个层次上的一次测试“微循环”。图中的其他流程可以是任意开发流程。例如，设计流程和编码流程。也可以是其他非开发流程，例如，SQA 流程，甚至是测试流程自身。也就是说，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以（或者说需要）进行了。

概括地说，H 模型揭示了：

- 软件测试不仅仅指测试的执行，还包括很多其他的活动。
- 软件测试是一个独立的流程，贯穿产品整个生命周期，与其他流程并发地进行。
- 软件测试要尽早准备，尽早执行。
- 软件测试是根据被测物的不同而分层次进行的。不同层次的测试活动可以是按照某个次序先后进行的，但也可能是反复的。

在 H 模型中，软件测试模型是一个独立的流程，贯穿于整个产品周期，与其他流程并发地进行。当某个测试时间点就绪时，软件测试即从测试准备阶段进入测试执行阶段。

2.6.4　其他模型

1．X 模型

由于 V 模型受到了很多人的质疑，因此，也有人提出了一些不同的观点和意见。在此，我们向大家介绍另外一种测试模型，即 X 模型，其目标是弥补 V 模型的一些缺陷。

X 模型的基本思想是由 Marick 提出的，但首先 Marick 不建议建立一个替代模型，同时，他也认为他的观点并不足以支撑一个模型的完整描述。不过，Robin F. Goldsmith 先生在自己的文章里将其思想定义为 X 模型，理由是，在 Marick 的观点中已经具备一个模型所需要的一些主要内容，其中也包括了像探索性测试这样的亮点。软件测试 X 模型如图 2-4 所示。

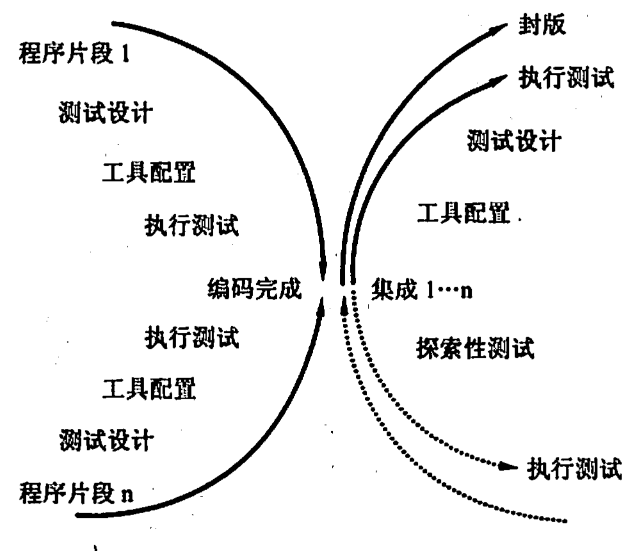


图 2-4 软件测试 X 模型

Marick 对 V 模型最主要的批评是 V 模型无法引导项目的全部过程。他认为一个模型必须能处理开发的所有方面，包括交接、频繁重复的集成以及需求文档的缺乏等。Marick 认为一个模型不应该规定那些和当前所公认的实践不一致的行为。

X 模型左边描述的是针对单独程序片段所进行的相互分离的编码和测试，此后，将进行频繁的交接，通过集成最终合成为可执行的程序。这一点在图的右上方得以体现，而且这些可执行程序还需要进行测试，已通过集成测试的成品可以进行封版并提交给用户，也可以作为更大规模和范围内集成的一部分。

同时，X 模型还定位了探索性测试，即如图 2-4 中右下方所示。这是不进行事先计划的特殊类型的测试，诸如“我这么测一下，结果会怎么样”，这一方式往往能帮助有经验的测试人员在测试计划之外发现更多的软件错误。

Marick 对 V 模型提出质疑，也是因为 V 模型是基于一套必须按照一定顺序严格排列的开发步骤，而这很可能并没有反映实际的实践过程。因为在实践过程中，很多项目是缺乏足够的需求的，而 V 模型还是从需求处理开始。

Marick 也质疑了单元测试和集成测试的区别，因为在某些场合人们可能会跳过单元测试而热衷于直接进行集成测试。Marick 担心人们盲目地跟随“学院派的 V 模型”，按照模型所指导的步骤进行工作，而实际上某些做法并不切合实用。

2．前置测试模型

前置测试模型是由 Robin F. Goldsmith 等人提出的，它是一个将测试和开发紧密结合的模型，该模型提供了轻松的方式，可以使你的项目加快速度。

前置测试模型如图 2-5 所示。

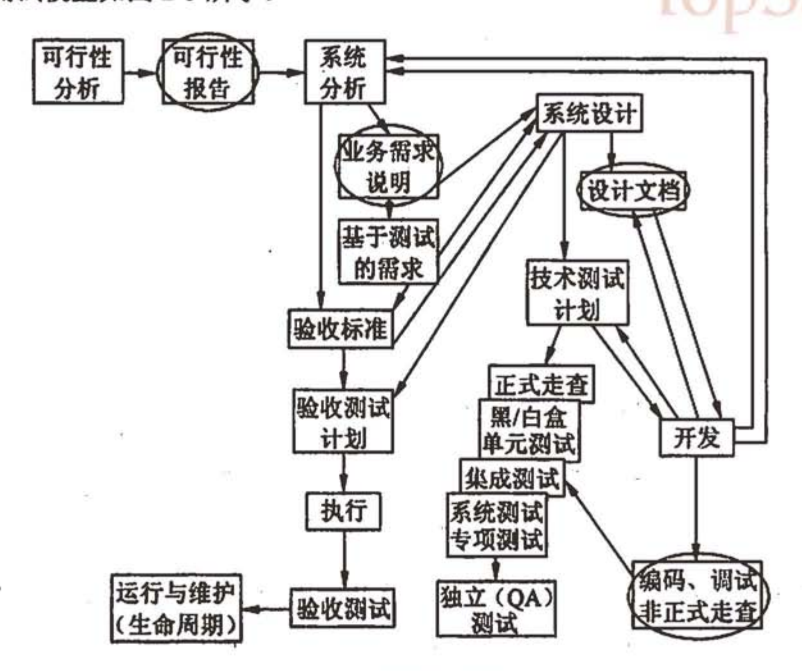


图 2-5 前置测试模型

前置测试模型体现了以下的要点。

- 开发和测试相结合：前置测试模型将开发和测试的生命周期整合在一起，标识了项目生命周期从开始到结束之间的关键行为。并且标识了这些行为在项目周期中的价值所在。如果其中有些行为没有得到很好的执行，那么项目成功的可能性就会因此而有所降低。如果有业务需求，则系统开发过程将更有效率。我们认为在没有业务需求的情况下进行开发和测试是不可能的。而且，业务需求最好在设计和开发之前就被正确定义。
- 对每一个交付内容进行测试：每一个交付的开发结果都必须通过一定的方式进行测试。源程序代码并不是惟一需要测试的内容。图中的椭圆框表示了其他一些要测试的对象，包括可行性报告、业务需求说明，以及系统设计文档等。这同 V 模型中开发和测试的对应关系是一致的，并且在其基础上有所扩展，变得更为明确。
- 在设计阶段进行测试计划和测试设计：设计阶段是作测试计划和测试设计的最好时机。很多组织要么根本不作测试计划和测试设计，要么在即将开始执行测试之前才飞快地完成测试计划和测试设计。在这种情况下，测试只是验证了程

序的正确性，而不是验证整个系统本该实现的东西。

- 测试和开发结合在一起：前置测试将测试执行和开发结合在一起，并在开发阶段以编码—测试—编码—测试的方式来体现。也就是说，程序片段一旦编写完成，就会立即进行测试。一般情况下，先进行的测试是单元测试，因为开发人员认为通过测试来发现错误是最经济的方式。但也可参考 X 模型，即一个程序片段也需要相关的集成测试，甚至有时还需要一些特殊测试。对于一个特定的程序片段，其测试的顺序可以按照 V 模型的规定，但其中还会交织一些程序片段的开发，而不是按阶段完全地隔离。
- 让验收测试和技术测试保持相互独立：验收测试应该独立于技术测试，这样可以提供双重的保险，以保证设计及程序编码能够符合最终用户的需求。验收测试既可以在实施阶段的第一步来执行，也可以在开发阶段的最后一步执行。前置测试模型提倡验收测试和技术测试沿循两条不同的路线来进行，每条路线分别地验证系统是否能够如预期设想的那样进行正常工作。这样，当单独设计好的验收测试完成了系统的验证时，我们即可确信这是一个正确的系统。

2.6.5 测试模型的使用

前面我们介绍了几种典型的测试模型，应该说这些模型对指导测试工作的进行具有重要的意义，但任何模型都不是完美的。我们应该尽可能地去应用模型中对项目有实用价值的方面，但不强行地为使用模型而使用模型，否则也没有实际意义。

在这些模型中，V 模型强调了在整个软件项目开发中需要经历的若干个测试级别，而且每一个级别都与一个开发级别相对应，但它忽略了测试的对象不应该仅仅包括程序，或者说它没有明确地指出应该对软件的需求、设计进行测试，而这一点在 W 模型中得到了补充。W 模型强调了测试计划等工作的先行和对系统需求和系统设计的测试，但 W 模型和 V 模型一样也没有专门针对软件测试的流程予以说明，因为事实上，随着软件质量要求越来越为大家所重视，软件测试也逐步发展成为一个独立于软件开发部的组织，就每一个软件测试的细节而言，它都有一个独立的操作流程。比如，现在的第三方测试，就包含了从测试计划和测试案例编写，到测试实施以及测试报告编写的全过程，这个过程在 H 模型中得到了相应的体现，表现为测试是独立的。也就是说，只要测试前提具备了，就可以开始进行测试了。当然，X 模型和前置测试模型又在此基础上增加了许多不确定因素的处理情况，因为在真实项目中，经常会有变更的发生，例如需要重新访问前一阶段的内容，或者跟踪并纠正以前提交的内容，修复错误，排除多余的成分，以及增加新发现的功能等。

因此，在实际的工作中，我们要灵活地运用各种模型的优点，在 W 模型的框架下，



运用 H 模型的思想进行独立地测试，并同时将测试和开发紧密结合，寻找恰当的就绪点开始测试并反复迭代测试，最终保证按期完成预定目标。

2.7 软件生命周期测试策略

2.7.1 软件开发与软件测试

在 W 模型中，我们对软件的需求、设计测试进行了阐述，本节重点强调软件生命周期中程序开发部分需要经历的若干个测试级别，采用 V 模型来进行介绍。

软件开发过程是一个自顶向下，逐步细化的过程。首先，在软件计划阶段定义了软件的作用域，然后进行软件需求分析，建立软件的数据域、功能和性能需求、约束及一些有效性准则。接着进入软件开发，进行软件设计，把设计用某种程序设计语言转换成程序代码。而测试过程则是依照相反的顺序安排自底向上，逐步集成的过程。低一级测试为上一级测试准备条件。当然不排除两者平行地进行测试。

如图 2-6 所示，首先对每一个程序模块进行单元测试，消除程序模块内部在逻辑上和功能上的错误和缺陷。再对照软件设计进行集成测试，检测和排除子系统（或系统）结构上的错误。随后再对照需求，进行确认测试。最后从系统整体出发，运行系统，看是否满足要求。

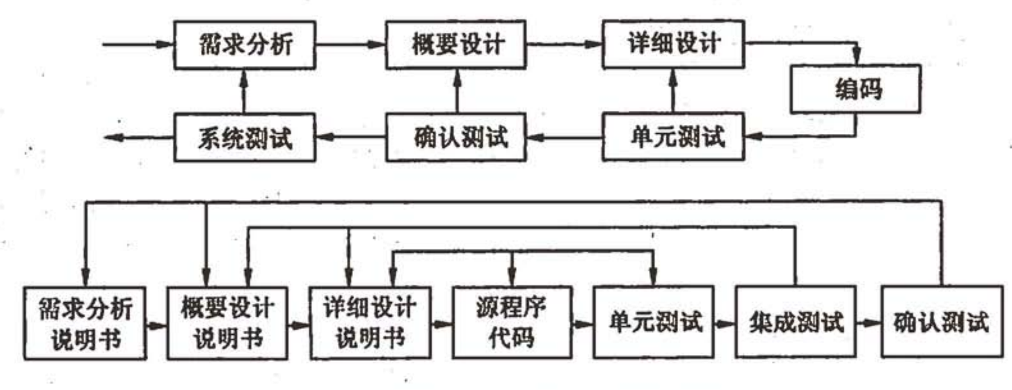


图 2-6 软件测试与软件开发过程的关系

2.7.2 软件测试策略

测试过程按 4 个步骤进行，即单元测试、集成（组装）测试、确认测试和系统测试。如图 2-7 所示显示出软件测试经历的 4 个步骤。

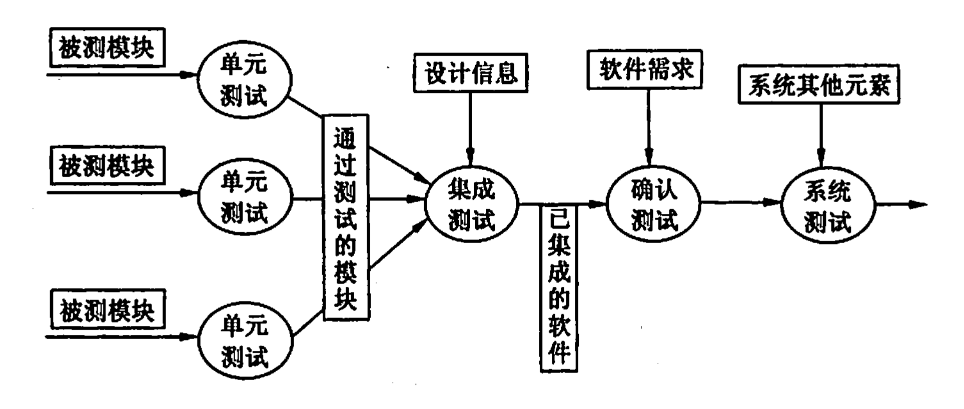


图 2-7　软件测试的过程

开始是单元测试，集中对用源代码实现的每一个程序单元进行测试，检查各个程序模块是否正确地实现了规定的功能。然后，把已测试过的模块组装起来，进行集成测试（组装测试），主要对与设计相关的软件体系结构的构造进行测试。为此，在将一个一个实施了单元测试并确保无误的程序模块组装成软件系统的过程中，对正确性和程序结构等方面进行检查。确认测试则是要检查已实现的软件是否满足了需求规格说明中确定了的各种需求，以及软件配置是否完全、正确。最后是系统测试，把已经经过确认的软件纳入实际运行环境中，与其他系统成分组合在一起进行测试。严格地说，系统测试已超出了软件工程的范围。

1．测试信息流

测试信息流如图 2-8 所示。测试过程需要以下三类输入。

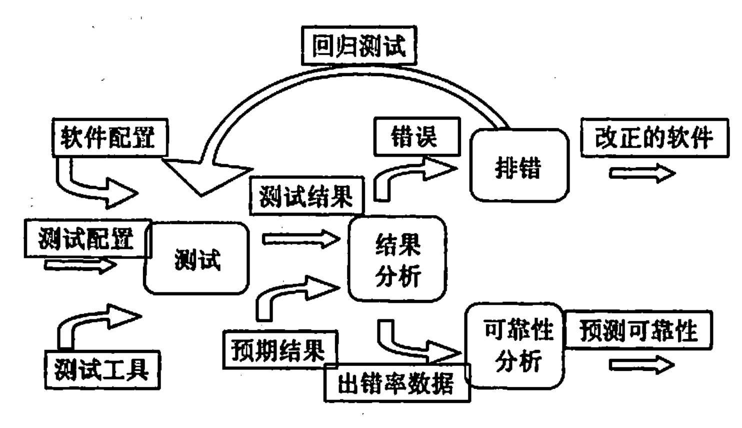


图 2-8　测试信息流



软件配置：包括软件需求规格说明、软件设计规格说明、源代码等。

测试配置：包括测试计划、测试用例、测试驱动程序等。实际上，在整个软件工程中，测试配置只是软件配置的一个子集。

测试工具：为提高软件测试效率，可使用测试工具支持测试工作，其作用就是为测试的实施提供某种服务，以减轻测试任务中的手工劳动。例如，测试数据自动生成程序、静态分析程序、动态分析程序、测试结果分析程序以及驱动测试的测试数据库等。

测试之后，要对所有测试结果进行分析，即将实测的结果与预期的结果进行比较。如果发现出错的数据，就意味着软件有错误，就需要开始排错（调试）。即对已经发现的错误进行错误定位和确定出错性质，并改正这些错误，同时修改相关的文档。修正后的文档一般都要经过再次测试，直到通过测试为止。

排错的过程是测试过程中最不可预知的部分，即使是一个与预期结果只相差 0.01% 的错误，也可能需要花上一个小时、一天、甚至一个月的时间去查找原因并改正错误。也正是因为排错中的这种固有的不确定性，使得我们很难确定可靠的测试进度。

通过收集和分析测试结果数据，开始针对软件建立可靠性模型。如果经常出现需要修改设计的严重错误，那么软件质量和可靠性就值得怀疑，同时也表明需要进一步测试。如果与此相反，软件功能能够正确完成，出现的错误易于修改，那么就可以断定：或者是软件的质量和可靠性达到可以接受的程度，或者是所作的测试不足以发现严重的错误。如果测试发现不了错误，那么几乎可以肯定，测试配置考虑得不够细致充分，错误仍然潜伏在软件中。这些错误最终不得不由用户在使用过程中发现，并在维护时由开发者去改正。但那时改正错误的费用将比在开发阶段改正错误的费用要高出 40～60 倍。

2．分析设计阶段

分析设计阶段的测试工作是评审与测试相结合的过程，主要包括需求说明书评测、概要设计说明书评测、详细设计说明书评测以及软件编码规范评测等。下述章节将详细论述。

（1）需求说明书评测

由于软件应用系统针对的行业广泛，因此在需求分析阶段可能存在着承建单位对业主单位的业务需求理解不全面、不准确的情况，常发生承建单位认为某一个业务功能的实现非常简单，而实际上业主单位业务标准的要求却很复杂的情况。在这种情况下，如果不通过评测进行相关的质量控制，往往造成承建单位按照自己的理解进行开发。如果不进行评测，或者评测之后没有充分发现问题，则给系统造成重大隐患，或者造成返工与延期。

因此，在此阶段评测的工作重点是与承建单位的分析人员、设计人员一起对需求说明书进行审查，并协调业主单位完成需求说明书的评审确认。

什么样的需求说明书是良好的，需求说明书编写应该遵照怎样的框架，针对需求说明书的评测有哪些主要内容等，这些在下述章节将详细论述。

- 编制良好的需求说明书 8 条原则。

1979 年由 Balzer 和 Goldman 提出了作出良好规格说明的 8 条原则。

原则 1：功能与实现分离，即描述要“做什么”而不是“怎样实现”。

原则 2：要求使用面向处理的规格说明语言，讨论来自环境的各种刺激可能导致系统做出什么样的功能性反应，来定义一个行为模型，从而得到“做什么”的规格说明。

原则 3：如果目标软件只是一个大系统中的一个元素，那么整个大系统也包括在规格说明的描述之中。描述该目标软件与系统的其他系统元素交互的方式。

原则 4：规格说明必须包括系统运行的环境。

原则 5：系统规格说明必须是一个认识的模型，而不是设计或实现的模型。

原则 6：规格说明必须是可操作的。规格说明必须是充分完全和形式的，以便能够利用它决定对于任意给定的测试用例，已提出的实现方案是否都能满足规格说明。

原则 7：规格说明必须容许不完备性并允许扩充。

原则 8：规格说明必须局部化和松散的耦合。它所包括的信息必须局部化，这样当信息被修改时，只要修改某个单个的段落（理想情况）。同时，规格说明应被松散地构造（即耦合），以便能够很容易地加入和删去一些段落。

尽管 Balzer 和 Goldman 提出的这 8 条原则主要用于基于形式化规格说明语言之上的需求定义的完备性，但这些原则对于其他各种形式的规格说明都适用。当然要结合实际来应用上述的原则。

- 需求说明书的框架。

需求说明书是分析任务的最终产物，通过建立完整的信息描述、详细的功能和行为描述、性能需求和设计约束的说明、合适的验收标准，给出对目标软件的各种需求。如表 2-1 中列出了需求说明书的框架。

表 2-1　需求说明书的框架

Ⅰ. 引言	A. 系统参考文献　B. 整体描述　C. 软件项目约束
Ⅱ. 信息描述	A. 信息内容表示　B. 信息流表示　i. 数据流　ii. 控制流
Ⅲ. 功能描述	A.功能划分　B.功能描述　i. 处理说明　ii. 限制/局限　iii. 性能需求 iv 设计约束　v. 支撑图　C.控制描述　i. 控制规格说明　ii. 设计约束
Ⅳ. 行为描述	A. 系统状态　B. 事件和响应
Ⅴ. 检验标准	A. 性能范围　B. 测试种类　C. 期望的软件响应　D. 特殊的考虑
Ⅵ. 参考书目	
Ⅶ. 附录	

- 需求说明书评测内容。

需求说明书评测作为需求分析阶段工作的复查手段，应该对功能的正确性、完整性和清晰性，以及其他需求给予评测。评测的主要内容是：

① 系统定义的目标是否与用户的要求一致；

② 系统需求分析阶段提供的文档资料是否齐全；

③ 文档中的所有描述是否完整、清晰，准确地反映用户要求；

④ 与所有其他系统成份的重要接口是否都已经描述；

⑤ 被开发项目的数据流与数据结构是否足够、确定；

⑥ 所有图表是否清楚，在不补充说明时能否理解；

⑦ 主要功能是否已包括在规定的软件范围之内，是否都已充分说明；

⑧ 软件的行为和它必须处理的信息、必须完成的功能是否一致；

⑨ 设计的约束条件或限制条件是否符合实际；

⑩ 是否考虑了开发的技术风险；

⑪ 是否考虑过软件需求的其他方案；

⑫ 是否考虑过将来可能会提出的软件需求；

⑬ 是否详细制定了检验标准，它们能否对系统定义是否成功进行确认；

⑭ 有没有遗漏、重复或不一致的地方；

⑮ 用户是否审查了初步的用户手册或原型；

⑯ 项目开发计划中的估算是否受到了影响。

为保证软件需求定义的质量，评测应由专门指定的人员负责，并按规程严格进行。评审结束，应有评审负责人的结论意见及签字。除承建单位分析员之外，业主单位人员和测试单位都应当参加评测工作。需求说明书要经过严格评测，一般，评测的结果都包括了一些修改意见，待修改完成后再经评测，才可进入设计阶段。根据上述讨论的评测内容，可以制定需求说明书评测规范，如表 2-2 所示。

填表说明：Y—是，TBD—不确定，N—否，NA—不适用。

表 2-2 需求说明书评测规范

编 号	评 测 项	评测结果 Y/TBD/N/NA
	清晰性	
1	系统的目标是否已定义	
2	是否对关键术语和缩略语进行定义和描述	
3	所使用的术语是否和用户/客户使用的一致	
4	需求的描述是否清晰，不含糊	

续表

编　　号	评　测　项	评测结果 Y/TBD/N/NA
	清晰性	
5	是否有对整套系统进行功能概述	
6	是否已详细说明了软件环境（共存的软件）和硬件环境（特定的配置）	
7	如果有会影响实施的假设情况，是否已经声明	
8	是否已经对每个业务逻辑进行输入、输出以及过程的详细说明	
	完整性	
9	是否列出了系统所必须的依赖、假设以及约束	
10	是否对每个提交物或阶段实施都进行了需求说明	
11	需求说明书是否已包括了主要的质量属性，例如有效性、高效性、灵活性、完整性、互操作性、可靠性、健壮性、可用性、可维护性、可移植性、可重用性和可测试性	
	依从性	
12	该文档是否遵守了该项目的文档编写标准	
	一致性	
13	需求说明是否存在直接相互矛盾的条目	
14	本需求说明书是否与相关需求素材一致	
	可行性	
15	所描述的所有功能是否必要并充分地满足客户/系统目标	
16	需求说明书描述的详细程度是否足以进行详细的设计	
17	已知的限制（局限）是否已经详细说明	
18	是否已确定每个需求的优先级别	
	可管理性	
19	是否将需求分别陈述，因此它们是独立的并且是可检查的	
20	是否所有需求都可以回溯到相应的需求素材，反之亦然	
21	是否已详细说明需求变更的过程	

在需求说明书评测结束后，测试单位应将评测意见以专题报告的形式提交业主单位。

（2）概要设计说明书评测

- 设计说明书的框架。如表2-3所示为软件设计规格说明的大纲。

软件设计的最终目标是要取得最佳方案。“最佳”是指在所有候选方案中，就节省开发费用，降低资源消耗，缩短开发时间的条件，选择能够赢得较高的生产率、较高的可靠性和可维护性的方案。在整个设计的过程中，各个时期的设计结果需要经过一系列设计质量的评测，以便及时发现和解决在软件设计中出现的问题，防止把问题遗留到开

发的后期阶段，造成后患。

表 2-3 软件设计规格说明大纲

Ⅰ. 工作范围 A. 系统目标 B. 运行环境 C. 主要软件需求 D. 设计约束/限制
Ⅱ. 体系结构设计 A. 数据流与控制流复审 B. 导出的程序结构 C. 功能与程序交叉索引
Ⅲ. 数据设计 A. 数据对象与形成的数据结构 B. 文件和数据库结构 i. 文件的逻辑结构 ii. 文件逻辑记录描述 iii. 访问方式 C. 全局数据 D. 文件/数据与程序交叉索引
Ⅳ. 接口设计 A. 人机界面规格说明 B. 人机界面设计规则 C. 外部接口设计 i. 外部数据接口 ii. 外部系统或设备接口 D. 内部接口设计规则
Ⅴ.（每个模块的）过程设计 A. 处理与算法描述 B. 接口描述 C. 设计语言(或其他)描述 D. 使用的模块 E. 内部程序逻辑描述 F. 注释/约束/限制
Ⅵ. 运行设计 A. 运行模块组合 B. 运行控制规则 C. 运行时间安排
Ⅶ. 出错处理设计 A. 出错处理信息 B. 出错处理对策 i. 设置后备 ii. 性能降级 iii. 恢复和再启动
Ⅷ. 安全保密设计
Ⅸ. 需求/设计交叉索引
Ⅹ. 测试部分 A. 测试方针 B. 集成策略 C. 特殊考虑
Ⅺ. 特殊注解
Ⅻ. 附录

- 概要设计说明书评测的内容。

① 可追溯性：即分析该软件的系统结构、子系统结构，确认该软件设计是否覆盖了所有已确定的软件需求，软件每一成份是否可追溯到某一项需求。

② 接口：即分析软件各部分之间的联系，确认该软件的内部接口与外部接口是否已经明确定义。模块是否满足高内聚和低耦合的要求。模块作用范围是否在其控制范围之内。

③ 风险：即确认该软件设计在现有技术条件下和预算范围内是否能按时实现。

④ 实用性：即确认该软件设计对于需求的解决方案是否实用。

⑤ 技术清晰度：即确认该软件设计是否以一种易于翻译成代码的形式表达。

⑥ 可维护性：从软件维护的角度出发，确认该软件设计是否考虑了方便未来的维护。

⑦ 质量：即确认该软件设计是否表现出良好的质量特征。

⑧ 各种选择方案：看是否考虑过其他方案，比较各种选择方案的标准是什么。

⑨ 限制：评估对该软件的限制是否现实，是否与需求一致。

⑩ 其他具体问题：对于文档、可测试性、设计过程等进行评估。

在这里需要特别注意：软件系统的一些外部特性的设计，例如软件的功能、一部分性能以及用户的使用特性等，在软件需求分析阶段就已经开始。这些问题的解决，多少

带有一些“怎么做”的性质，因此有人称之为软件的外部设计。

为评测设计是否达到目标，必须建立衡量设计的技术标准。如下：

① 设计出来的结构应是分层结构，从而建立软件成分之间的控制。

② 设计应当模块化，从逻辑上将软件划分为完成特定功能或子功能的构件。

③ 设计应当既包含数据抽象，也包含过程抽象。

④ 设计应当建立具有独立功能特征的模块。

⑤ 设计应当建立能够降低模块与外部环境之间复杂连接的接口。

⑥ 设计应能根据软件需求分析获取的信息，建立可驱动、可重复的方法。

根据上述讨论的评测内容以及评测标准，可以建立概要设计说明书评测规范，如表2-4所示。

填表说明：Y—是，TBD—不确定，N—否，NA—不适用。

表2-4　概要设计说明书评测规范

编号	评　测　项	评测结果 Y/TBD/N/NA
	清晰性	
1	是否所设计的架构，包括数据流、控制流和接口，被清楚地表达了	
2	是否所有的假设、约束、策略及依赖都被记录在本文档了	
3	是否定义了总体设计目标	
	完整性	
4	是否所有的以前的TBD（待确定条目）都已经被解决了	
5	是否设计已经可以支持本文档中遗留的TBD有可能带来的变更	
6	是否所有的TBD的影响都已经被评估了	
7	是否仍存在可能不可行的设计部分	
8	是否已记录设计时的权衡考虑，该文件是否包括了权衡选择的标准和不选择其他方案的原因	
	依从性	
9	是否遵守了项目的文档编写标准	
	一致性	
10	数据元素、流程和对象的命名和使用在整套系统和外部接口之间是否一致	
11	该设计是否反映了实际操作环境（硬件、软件和支持软件）	
	可行性	
12	从进度、预算和技术角度上看该设计是否可行	
13	是否存在错误的、缺少的或不完整的逻辑	
	数据使用	
14	所有复合数据元素、参数以及对象的概念是否都已文档化	



续表

编号	评 测 项	评测结果 Y/TBD/N/NA
	数据使用	
15	是否还有任何需要的，但还没有定义的数据结构，反之亦然	
16	是否已描述最低级别数据元素，是否已详细说明取值范围	
	功能性	
17	是否对每一下级模块进行了概要算法说明	
18	所选择的设计和算法能否满足所有的需求	
	接口	
19	操作界面的设计是否有为用户考虑（例如：词汇、使用信息和进入的简易）	
20	是否已描述界面的功能特性	
21	界面将有利于问题解决吗	
22	是否所有界面都互相一致，与其他模块一致，以及和更高级别文档中的需求一致	
23	是否所有的界面都提供了所要求的信息	
24	是否已说明内部各界面之间的关系	
25	界面的数量和复杂程度是否已减少到最小	
	可维护性	
26	该设计是否是模块化的	
27	这些模块具有高内聚度和低耦合度吗	
28	是否已经对继承设计、代码或先前选择工具的使用进行了详细说明性能	
29	主要性能参数是否已被详细说明（例如：实时、速度要求、磁盘输入/输出接口等）	
	可靠性	
30	该设计能够提供错误检测和恢复吗（例如：输入输出检查）	
31	是否已考虑非正常情况	
32	是否所有的错误情况都被完整并准确地说明	
33	该设计是否满足该系统进行集成时所遵守的约定	
	易测性	
34	是否能够对该套系统进行测试、演示、分析或检查来说明它是满足需求的	
35	该套系统是否能用增量型的方法来集成和测试	
	可追溯性	
36	是否各部分的设计都能追溯到需求说明书的需求	
37	是否所有的设计决策都能追溯到原来确定的权衡因素	
38	所继承设计的已知风险是否已确定和分析	

（3）详细设计说明书评测

详细设计说明书的评测标准和评测内容与概要设计说明书基本相同，这里不再赘述。如表 2-5 所示为详细设计说明书评测规范。

填表说明：Y—是，TBD—不确定，N—否，NA—不适用。

表 2-5　详细设计说明书评测规范

编号	评　测　项	评测结果 Y/TBD/N/NA
	清晰性	
1	所有单元或过程的目的是否都已文档化	
2	包括了数据流、控制流和接口的单元设计是否已清晰地说明	
	完整性	
3	是否已定义和初始化所有的变量、指针和常量	
4	是否已描述单元的全部功能	
5	是否已详细说明用来实现该单元的关键算法（例如：用自然语言或 PDL）	
6	是否已列出该单元的调用	
	依从性	
7	该文档是否遵循了该项目已文档化的标准	
8	是否采用了所要求的方法和工具来进行单元设计	
	一致性	
9	数据元素的命名和使用在整个单元和单元接口之间是否一致	
10	所有接口的设计是否互相一致并且和更高级别文档一致	
	正确性	
11	是否处理所有条件（>0、=0、<0、switch/case），是否存在处理“case not found”的条件	
12	是否正确地规定了分支（逻辑没有颠倒）	
	数据使用	
13	是否所有声明的数据都被实际使用到	
14	是否所有该单元的数据结构都被详细说明	
15	是否所有修改共享数据（或文件）的程序都考虑到了其他程序对该共享数据（或文件）的存取权限	
16	是否所有逻辑单元、时间标志和同步标志都被定义和初始化	
	接口	
17	接口参数在数量、类型和顺序上是否匹配	
18	是否所有的输入和输出都被正确定义和检查	
19	是否传递参数序列都被清晰地描述	
20	是否所有参数和控制标志由已描述的单元传递或返回	



续表

编号	评　测　项	评测结果 Y/TBD/N/NA
	接口	
21	是否详细说明了参数的度量单位、取值范围、正确度和精度	
22	共享数据区域及其存取规定的映射是否一致	
	可维护性	
23	单元是否具有高内聚度和低耦合度（例如：对该单元的更改不会在该单元有任何无法预料的影响并对其他单元的影响很小）	
	性能	
24	是否该单元的所有约束（例如：过程时间和规模）都被详细说明	
	可靠性	
25	初始化是否使用到缺省值，缺省值是否正确	
26	是否在内存访问的时候执行了边界检查（例如：数组、数据结构、指针等）来确保只是改变了目标存储位置	
27	是否执行输入、输出、接口和结果的错误检查	
28	是否对所有错误情况都发出有意义的信息	
29	对特殊情况返回的代码是否和已规定的全局定义的返回代码相匹配	
30	是否考虑到意外事件	
	易测性	
31	是否能够对每个单元进行测试、演示、分析或检查来说明它们是满足需求的	
32	该设计是否包含检查点来帮助测试（例如：有条件的编译代码和数据声明测试）	
33	是否所有的逻辑都能被测试	
34	是否已描述测试程序、测试数据集和测试结果	
	可追溯性	
35	是否设计的每一部分都能追溯到其他项目文档的需求，也能追溯到更高级别文档的需求	
36	是否所有的设计决定都能追溯到权衡考虑	
37	单元需求是否都能上溯到更高级别的文档，更高级别文档的需求是否已经在单元中体现	

（4）软件编码规范评测

程序实际上也是一种供人阅读的文章，有一个文章的风格问题。程序良好的风格表现在源程序文档化、数据说明的方法、语句结构和输入/输出方法这四个方面，软件编码规范评测也是围绕这四个方面展开。下面分别论述评测内容以及相应的评测标准。

- 源程序文档化。

① 符号名的命名。符号名即标识符，包括模块名、变量名、常量名、标号名、子程序名、数据区名以及缓冲区名等。这些名字应能反映它所代表的实际东西，应有一定实际意义。例如，表示次数的量用 Times，表示总量的用 Total，表示平均值的用 Average，表示和的量用 Sum 等。

名字不是越长越好，应当选择精炼的、意义明确的名字。必要时可使用缩写名字，但这时要注意缩写规则要一致，并且要给每一个名字加注释。同时，在一个程序中，一个变量只应用于一种用途。

② 程序的注释。夹在程序中的注释是程序员日后与程序读者之间通信的重要手段。注释绝不是可有可无的。一些正规的程序文本中，注释行的数量占到整个源程序的 1/3～1/2，甚至更多。注释分为序言性注释和功能性注释。

序言性注释通常置于每个程序模块的开头部分，它应当给出程序的整体说明，对于理解程序本身具有引导作用。有些软件开发部门对序言性注释做了明确而严格的规定，要求程序编制者逐项列出。有关项目包括：程序标题；有关本模块功能和目的的说明；主要算法；接口说明：包括调用形式，参数描述，子程序清单；有关数据描述：重要的变量及其用途，约束或限制条件，以及其他有关信息；模块位置：在哪一个源文件中，或隶属于哪一个软件包；开发简历：模块设计者，复审者，复审日期，修改日期及有关说明等。

功能性注释嵌在源程序体中，用以描述其后的语句或程序段是在做什么工作，或是执行了下面的语句会怎么样。而不要解释下面怎么做。要点：描述一段程序，而不是每一个语句；用缩进和空行，使程序与注释容易区别；注释要正确。

③ 标准的书写格式。视觉组织用空格、空行和移行来实现。恰当地利用空格，可以突出运算的优先性，减少编码的错误；自然的程序段之间可用空行隔开；移行也叫做向右缩格。它是指程序中的各行不必都在左端对齐，都从第一格起排列，这样做使程序完全分不清层次关系。对于选择语句和循环语句，把其中的程序段语句向右作阶梯式移行。使程序的逻辑结构更加清晰。

- 数据说明。

在设计阶段已经确定了数据结构的组织及其复杂性。在编写程序时，则需要注意数据说明的风格。为了使程序中数据说明更易于理解和维护，必须注意以下几点。

① 数据说明的次序应当规范化。数据说明次序规范化，使数据属性容易查找，也有利于测试，排错和维护。原则上，数据说明的次序与语法无关，其次序是任意的。但出于阅读、理解和维护的需要，最好使其规范化，使说明的先后次序固定。

② 说明语句中变量安排有序化。当多个变量名在一个说明语句中说明时，应当对这些变量按字母的顺序排列。带标号的全程数据也应当按字母的顺序排列。



③ 使用注释说明复杂数据结构。如果设计了一个复杂的数据结构，应当使用注释来说明在程序实现时这个数据结构的固有特点。

- 语句结构。

在设计阶段确定了软件的逻辑流结构，但构造单个语句则是编码阶段的任务。语句构造力求简单、直接，不能为了片面追求效率而使语句复杂化。

比如：在一行内只写一条语句；程序编写首先应当考虑清晰性；程序要能直截了当地说明程序员的用意；除非对效率有特殊的要求，程序编写要做到清晰第一，效率第二，不要为了追求效率而丧失了清晰性；首先要保证程序正确，然后才要求提高速度，反过来说，在使程序高速运行时，首先要保证它是正确的；避免使用临时变量而使可读性下降；对编译程序做简单的优化；尽可能使用库函数；避免不必要的转移；尽量采用基本的控制结构来编写程序；避免采用过于复杂的条件测试；尽量减少使用“否定”条件的条件语句；尽可能用通俗易懂的伪码来描述程序的流程，然后再翻译成必须使用的语言；数据结构要有利于程序的简化；程序要模块化，使模块功能尽可能单一化，模块间的耦合能够清晰可见；利用信息隐蔽，确保每一个模块的独立性；从数据出发去构造程序；不要修补不好的程序，要重新编写。

- 输入和输出

输入和输出信息是与用户的使用直接相关的。输入和输出的方式和格式应当尽可能方便用户的使用。一定要避免因设计不当给用户带来的麻烦。因此，在软件需求分析阶段和设计阶段，就应基本确定输入和输出的风格。系统能否被用户接受，有时就取决于输入和输出的风格。输入/输出风格还受到许多其他因素的影响。如输入/输出设备（终端的类型，图形设备，数字化转换设备等）、用户的熟练程度以及通信环境等。不论是批处理的输入/输出方式，还是交互式的输入/输出方式，在设计和程序编码时都应考虑下列原则。

① 对所有的输入数据都要进行检验，识别错误的输入，以保证每个数据的有效性；

② 检查输入项的各种重要组合的合理性，必要时报告输入状态信息；

③ 使输入的步骤和操作尽可能简单，并保持简单的输入格式；

④ 输入数据时，应允许使用自由格式输入；

⑤ 应允许缺省值；

⑥ 输入一批数据时，最好使用输入结束标志，而不要由用户指定输入数据数目；

⑦ 在交互式输入时，要在屏幕上使用提示符，明确提示交互输入的请求，指明可使用选择项的种类和取值范围。同时，在数据输入的过程中和输入结束时，也要在屏幕上给出状态信息；

⑧ 当程序设计语言对输入/输出格式有严格要求时，应保持输入格式与输入语句要

求的一致性；

⑨ 给所有的输出加注解，并设计输出报表格式。

3．开发阶段

（1）单元测试

单元测试又称模块测试，是针对软件设计的最小单位——程序模块，进行正确性检验的测试工作。其目的在于发现各模块内部可能存在的各种差错。单元测试需要从程序的内部结构出发设计测试用例。多个模块可以平行地独立进行单元测试。

- 单元测试的内容。

在进行单元测试时，测试者需要依据详细设计说明书和源程序清单，了解该模块的 I/O 条件和模块的逻辑结构，主要采用白盒测试的测试用例，辅之以黑盒测试的测试用例，使之对任何合理的输入和不合理的输入，都能鉴别和响应。这要求对所有的局部的和全局的数据结构、外部接口和程序代码的关键部分，都要进行桌面检查和严格的代码审查。

在单元测试中进行的测试工作如图 2-9 所示，需要在五个方面对所测模块进行检查。

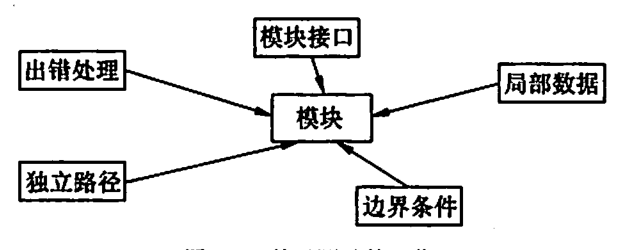


图 2-9　单元测试的工作

① 模块接口测试。

在单元测试的开始，应对通过所测模块的数据流进行测试。如果数据不能正确地输入和输出，就谈不上进行其他测试。为此，对模块接口可能需要如下的测试项目：调用所测模块时的输入参数与模块的形式参数在个数、属性、顺序上是否匹配；所测模块调用子模块时，它输入给子模块的参数与子模块中的形式参数在个数、属性、顺序上是否匹配；是否修改了只作输入用的形式参数；输出给标准函数的参数在个数、属性、顺序上是否正确；全局量的定义在各模块中是否一致；限制是否通过形式参数来传送。

当模块通过外部设备进行输入/输出操作时，必须附加如下的测试项目：文件属性是否正确；OPEN 语句与 CLOSE 语句是否正确；规定的 I/O 格式说明与 I/O 语句是否匹配；缓冲区容量与记录长度是否匹配；在进行读写操作之前是否打开了文件；在结束文件处理时是否关闭了文件；正文书写/输入错误，以及 I/O 错误是否检查并做了处理。



② 局部数据结构测试。

模块的局部数据结构是最常见的错误来源，应设计测试用例以检查以下各种错误：不正确或不一致的数据类型说明；使用尚未赋值或尚未初始化的变量；错误的初始值或错误的缺省值；变量名拼写错或书写错；不一致的数据类型。可能的话，除局部数据之外的全局数据对模块的影响也需要查清。

③ 路径测试。

由于通常不可能做到穷举测试，所以在单元测试期间要选择适当的测试用例，对模块中重要的执行路径进行测试。应当设计测试用例查找由于错误的计算、不正确的比较或不正常的控制流而导致的错误。对基本执行路径和循环进行测试，可以发现大量的路径错误。

常见的不正确计算有：运算的优先次序不正确或误解了运算的优先次序；运算的方式错，即运算的对象彼此在类型上不相容；算法错；初始化不正确；运算精度不够；表达式的符号表示不正确。

常见的比较和控制流错误有：不同数据类型的相互比较；不正确的逻辑运算符或优先次序；因浮点数运算精度问题而造成的两值比较不等；关系表达式中不正确的变量和比较符；“差 1”错，即不正确地多循环一次或少循环一次；错误的或不可能的循环中止条件；当遇到发散的迭代时不能中止的循环；不适当地修改了循环变量等。

④ 错误处理测试。

比较完善的模块设计要求能预见出错的条件，并设置适当的出错处理，以便在一旦程序出错时，能对出错程序重做安排，保证其逻辑上的正确性。这种出错处理也应当是模块功能的一部分。若出现下列情况之一，则表明模块的错误处理功能包含有错误或缺陷：出错的描述难以理解；出错的描述不足以对错误定位，不足以确定出错的原因；显示的错误与实际的错误不符；对错误条件的处理不正确；在对错误进行处理之前，错误条件已经引起系统的干预等。

⑤ 边界测试。

在边界上出现错误是常见的。例如，在一段程序内有一个 n 次循环，当到达第 n 次重复时就可能会出错。另外，在取最大值或最小值时也容易出错。因此，要特别注意数据流、控制流中刚好等于、大于或小于确定的比较值时出错的可能性。对这些地方要仔细地选择测试用例，认真加以测试。

此外，如果对模块运行时间有要求的话，还要专门进行关键路径测试，以确定最坏情况下和平均意义下影响模块运行时间的因素。这类信息对进行性能评价是十分有用的。

虽然模块测试通常是由编写程序的人自己完成的，但是项目负责人应当关心测试的结果。所有测试用例和测试结果都是模块开发的重要资料，必须妥善保存。

总之，模块测试针对的程序规模较小，易于查错；发现错误后容易确定错误的位置，易于排错，同时多个模块可以并行测试。做好模块测试可为后续的测试打下良好的基础。

- 单元测试的步骤。

通常单元测试是在编码阶段进行的。在源程序代码编制完成，经过评审和验证，确认没有语法错误之后，就开始进行单元测试的测试用例设计。利用设计文档，设计可以验证程序功能、找出程序错误的多个测试用例。对于每一组输入，应有预期的正确结果。

模块并不是一个独立的程序，在考虑测试模块时，同时要考虑它和外界的联系，用一些辅助模块去模拟与所测模块相联系的其他模块。这些辅助模块分为两种：

驱动模块（driver）——相当于所测模块的主程序。它接收测试数据，把这些数据传送给所测模块，最后再输出实测结果。

桩模块（stub）——也叫做存根模块。用以代替所测模块调用的子模块。桩模块可以做少量的数据操作，不需要把子模块所有功能都带进来，但不允许什么事情也不做。

所测模块、与它相关的驱动模块及桩模块共同构成了一个“测试环境”，如图 2-10 所示。驱动模块和桩模块的编写会给测试带来额外的开销。因为它们在软件交付时不作为产品的一部分一同交付，而且它们的编写需要一定的工作量。特别是桩模块，不能只简单地给出“曾经进入”的信息。为了能够正确地测试软件，桩模块可能需要模拟实际子模块的功能，这样，桩模块的建立就不是很轻松了。

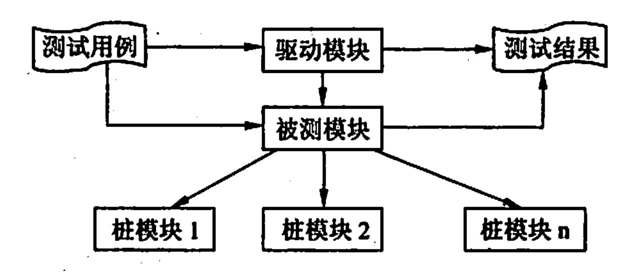


图 2-10　单元测试的测试环境

模块的内聚程度高，可以简化单元测试过程。如果每一个模块只完成一种功能，则需要的测试用例数目将明显减少，模块中的错误也容易被预测和发现。

当然，如果一个模块要完成多种功能，且以程序包（package）的形式出现的也不少见，这时可以将这个模块看成由几个小程序组成。必须对其中的每个小程序先进行单元测试要做的工作，对关键模块还要做性能测试。对支持某些标准规程的程序，更要着手进行互联测试。有人把这种情况特别称为模块测试，以区别单元测试。

（2）集成测试

集成测试也叫做组装测试或联合测试。通常，在单元测试的基础上，需要将所有模



块按照概要设计说明书和详细设计说明书的要求进行组装。

- 组装时需要考虑的问题。

① 在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失；

② 一个模块的功能是否会对另一个模块的功能产生不利的影响；

③ 各个子功能组合起来，能否达到预期要求的父功能；

④ 全局数据结构是否有问题；

⑤ 单个模块的误差累积起来，是否会放大，以至达到不能接受的程度。

因此，在单元测试的同时可进行集成测试，发现并排除在模块连接中可能出现的问题，最终构成要求的软件系统。

子系统的集成测试称为部件测试，它所做的工作是要找出组装后的子系统与系统需求规格说明之间的不一致。

选择什么方式把模块组装起来形成一个可运行的系统，直接影响到模块测试用例的形式、所用测试工具的类型、模块编号的次序和测试的次序以及生成测试用例的费用和调试的费用。

- 模块组装成为系统的方式。

模块组装成为系统的方式有两种：一次性组装方式和增殖式组装方式。

① 一次性组装方式（big bang）。

它是一种非增殖式组装方式，也叫做整体拼装。使用这种方式，首先对每个模块分别进行模块测试，再把所有模块组装在一起进行测试，最终得到要求的软件系统。例如，有一个模块系统结构，如图 2-11（a）所示。其单元测试和组装顺序如图 2-11（b）所示。

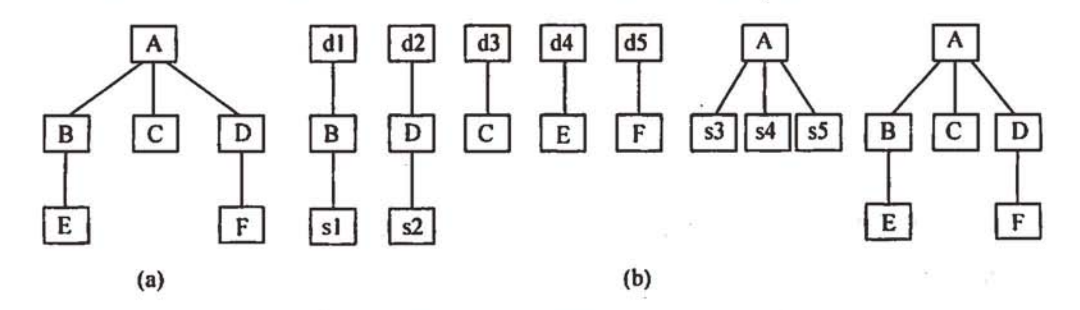


图 2-11 一次性组装方式

在如图 2-11（b）中，模块 d1，d2，d3，d4，d5 是对各个模块做单元测试时建立的驱动模块，s1，s2，s3，s4，s5 是为单元测试而建立的桩模块。这种一次性组装方式试图在辅助模块的协助下，在分别完成模块单元测试的基础上，将所测模块连接起来进行测试。但是由于程序中不可避免地存在涉及模块间接口、全局数据结构等方面的问题，

所以一次试运行成功的可能性并不很大。其结果是，发现有错误，却茫然找不到原因。查错和改错都会遇到困难。

② 增殖式组装方式。

这种组装方式又称渐增式组装，是首先对一个个模块进行模块测试，然后将这些模块逐步组装成较大的系统，在组装的过程中边连接边测试，以发现连接过程中产生的问题。最后通过增殖逐步组装成为要求的软件系统。

- 自顶向下的增殖方式。这种组装方式是将模块按系统程序结构，沿控制层次自顶向下进行组装。其步骤如下：首先以主模块作为所测模块兼驱动模块，所有直属于主模块的下属模块全部用桩模块代替，对主模块进行测试。再采用深度优先（如图 2-12 所示为自顶向下的增殖方式）或广度优先的策略，用实际模块替换相应的桩模块，再用桩模块代替它们的直接下属模块，与已测试的模块或子系统组装成新的子系统。然后，进行回归测试（即重新执行以前做过的全部测试或部分测试），排除组装过程中引入新的错误的可能。最后，判断是否所有的模块都已组装到系统中。是，则结束测试；否则，转到 B 去执行。

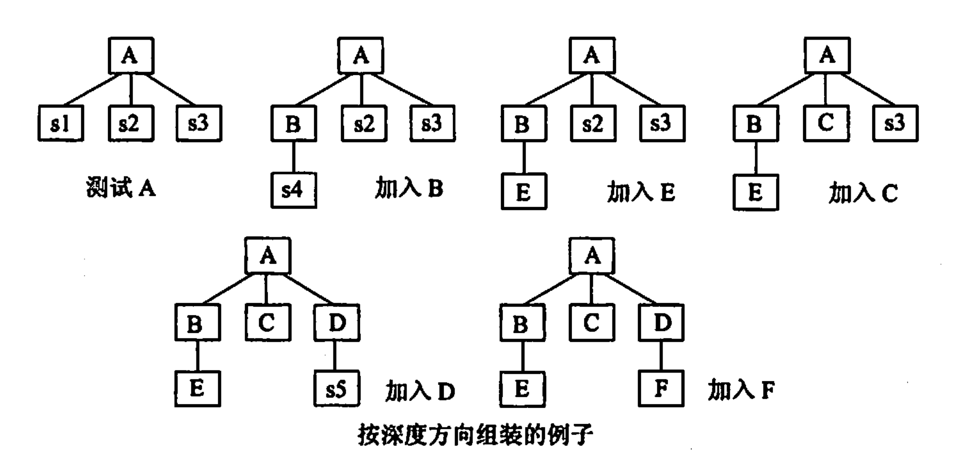


图 2-12　自顶向下的增殖方式

自顶向下的增殖方式在测试过程中较早地验证了主要的控制和判断点。在一个功能划分合理的程序模块结构中，判断常常出现在较高的层次里，因而，能够较早地遇到这种问题。如果主要控制有问题，尽早发现它能够减少以后的返工，这是十分必要的。如果选用按深度方向组装的方式，可以首先实现和验证一个完整的软件功能，可先对逻辑输入的分支进行组装和测试，检查和克服潜藏的错误和缺陷，验证其功能的正确性，就为其后对主要加工分支的组装和测试提供了保证。此外，功能可行性较早地得到证实，

还能够增强开发者和用户成功的信心。

- 自底向上的增殖方式。这种组装方式是从程序模块结构的最底层模块开始组装和测试。因为模块是自底向上进行组装的，对于一个给定层次的模块，它的子模块（包括子模块的所有下属模块）已经组装并测试完成，所以不再需要桩模块。在模块的测试过程中需要从子模块得到的信息可以通过直接运行子模块得到。自底向上增殖的步骤如下：首先由驱动模块控制最底层模块的并行测试；也可以把最底层模块组合成实现某一特定软件功能的簇，由驱动模块控制它进行测试。再用实际模块代替驱动模块，与它已测试的直属子模块组装成为子系统。然后，为子系统配备驱动模块，进行新的测试。最后判断是否已组装到达主模块。是，则结束测试；否则，执行 B。

以如图 2-11（a）所示的一次性组装方式系统结构为例，可以用如图 2-13 说明自底向上组装和测试的顺序。

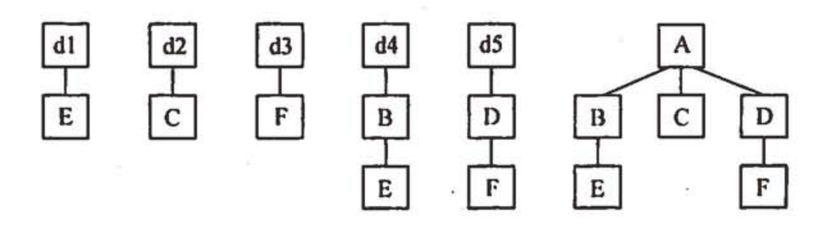


图 2-13 自底向上的增殖方式

- 混合增殖式测试。自顶向下增殖的方式和自底向上增殖的方式各有优缺点。一般来讲，一种方式的优点是另一种方式的缺点。

自顶向下增殖方式的缺点是需要建立桩模块。要使桩模块能够模拟实际子模块的功能十分困难，因为，桩模块在接收了所测模块发送的信息后，需要按照它所代替的实际子模块功能返回应该回送的信息，这必将增加建立桩模块的复杂度，而且导致增加一些附加的测试。同时，涉及复杂算法和真正输入/输出的模块一般在底层，它们是最容易出问题的模块，到组装和测试的后期才遇到这些模块，一旦发现问题，就会导致过多的回归测试。而自顶向下增殖方式的优点是能够较早地发现主要控制方面的问题。

自底向上增殖方式的缺点是“程序一直未能作为一个实体存在，直到最后一个模块加上去后才形成一个实体”。就是说，在自底向上组装和测试的过程中，对主要的控制直到最后才接触到。这种方式的优点是不需要桩模块，而建立驱动模块一般比建立桩模块容易，同时由于涉及到复杂算法和真正输入/输出的模块最先得到组装和测试，可以把最容易出问题的部分在早期解决。此外自底向上增殖的方式可以实施多个模块的并行测试，

提高测试效率。因此，通常是把以上两种方式结合起来进行组装和测试。

在进行集成测试时，测试者应当确定关键模块，对这些关键模块及早进行测试。关键模块至少应具有以下几种特征之一：

- 满足某些软件需求；
- 在程序的模块结构中位于较高的层次（高层控制模块）；
- 较复杂、较易发生错误；
- 有明确定义的性能要求。

在做回归测试时，也应该集中测试关键模块的功能。

- 集成测试的组织和实施。

集成测试是一种正规测试过程，必须精心计划，并与单元测试的完成时间协调起来。在制定测试计划时，应考虑如下因素：

① 采用何种系统组装方法来进行集成测试。

② 集成测试过程中连接各个模块的顺序。

③ 模块代码编制和测试进度是否与集成测试的顺序一致。

④ 测试过程中是否需要专门的硬件设备。

解决了上述问题之后，就可以列出各个模块的编制、测试计划表，标明每个模块单元测试完成的日期、首次集成测试的日期、集成测试全部完成的日期、以及需要的测试用例和所期望的测试结果。

在缺少软件测试所需要的硬件设备时，应检查该硬件的交付日期是否与集成测试计划一致。例如，若测试需要数字化仪和绘图仪，则相应的测试应安排在这些设备能够投入使用之时，并要为硬件的安装和交付使用保留一段时间，以留下时间余量。此外，在测试计划中需要考虑测试所需软件（驱动模块、桩模块、测试用例生成程序等）的准备情况。

- 集成测试完成的标志。

集成测试完成的标志主要有以下几项。

① 成功地执行了测试计划中规定的所有集成测试。

② 修正了所发现的错误。

③ 测试结果通过了专门小组的评审。

集成测试应由专门的测试小组来进行，测试小组由有经验的系统设计人员和程序员组成。整个测试活动要在评审人员出席的情况下进行。

在完成预定的集成测试工作之后，测试小组应负责对测试结果进行整理、分析，形成测试报告。测试报告中要记录实际的测试结果在测试中发现的问题、解决这些问题的方法以及解决之后再次测试的结果。此外还应提出目前不能解决、还需要管理人员和开

发人员注意的一些问题，提供测试评审和最终决策，以提出处理意见。

集成测试需要提交的文档有集成测试计划、集成测试规格说明和集成测试分析报告。

（3）确认测试

确认测试的任务是验证软件的功能和性能及其他特性是否与用户的要求一致。对软件的功能和性能要求在软件需求规格说明中明确规定。确认测试一般包括有效性测试和软件配置复查，确认测试一般由独立的第三方测试机构进行。

- 进行有效性测试。

有效性测试是在模拟的环境下，运用黑盒测试的方法，验证所测软件是否满足需求规格说明书列出的需求。为此，需要制定测试计划、测试步骤以及具体的测试用例。通过实施预定的测试计划和测试步骤，确定软件的特性是否与需求相符，确保所有的软件功能需求都能得到满足，所有的软件性能需求都能达到。所有的文档都是正确且便于使用的。同时，对其他软件需求，例如可移植性、可靠性、易用性、兼容性、可维护性等，也都要进行测试，确认是否满足。

在全部软件测试的测试用例运行完后，所有的测试结果可以分为两类。

① 测试结果与预期的结果相符。这说明软件的这部分功能或性能特征与需求规格说明书相符合，从而接受了这部分程序。

② 测试结果与预期的结果不符。这说明软件的这部分功能或性能特征与需求规格说明不一致，因此要为它提交一份问题报告。

- 软件配置复查。

软件配置复查的目的是保证软件配置的所有成分都齐全，各方面的质量都符合要求，具有维护阶段所必须的细节，而且已经编排好分类的目录。

在确认测试的过程中，还应当严格遵守用户手册和操作手册中规定的使用步骤，以便检查文档资料的完整性和正确性。

（4）系统测试

系统测试是将通过集成测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际或者模拟运行（使用）环境下，对计算机系统进行一系列测试。

系统测试的目的在于通过与系统的需求定义作比较，发现软件与系统定义不符合或与之矛盾的地方。

（5）验收测试

验收测试是以用户为主的测试。软件开发人员和质量保证人员也应参加。由用户参加设计测试用例。使用用户界面输入测试数据，并分析测试的输出结果。一般使用生产中的实际数据进行测试。

目前在国内实际软件开发，特别是系统集成的过程中，验收测试往往在系统测试完成后、项目最终交付前进行。验收测试的测试计划、测试方案与测试案例一般由开发方制定，由用户方与监理方联合进行评审。验收小组由开发方、用户方、监理方代表、主管单位领导及行业专家构成。与确认测试及系统测试不同的是，验收测试往往不是对系统的全覆盖测试，而是针对用户的核心业务流程进行的测试；同时，测试的执行人员也不是开发方的测试组成员，而是由用户方的使用人员完成。

近年来，越来越多的开发方及用户方认识到对项目进行最终验收测试的重要意义，因此，由第三方完成的专业化全覆盖型技术测试得到了广泛应用。由专门从事测试工作的第三方机构，根据系统的需求分析、用户手册、培训手册等，在开发人员及最终使用人员的配合下，完成对系统全面的测试工作。

4．软件验证与确认（V&V）过程

软件的验证与确认（V&V）是贯穿软件生命周期的重要的质量保证过程，国际标准化组织 IEEE 在 1886 年颁布了软件 V&V 标准，又于 1998 年修订颁布了 IEEE/ANSI Std 1012-1998 软件验证与确认计划。标准规定了软件验证和确认过程（简称 V&V）和软件验证和确认计划（简称 SVVP）编制要求。我国软件验证与确认（V&V）的国家标准也即将颁布实施，将在我国软件的质量保证和软件测试的工作中发挥重要作用。

软件的 V&V 过程是确定按照规定的软件过程开发的产品是否符合活动的要求，软件是否满足它的预期用途和用户需要。软件的 V&V 过程包括软件产品和过程的分析、评价、评审、审核、评估和测试。

软件测试活动是软件 V&V 过程的一个组成部分。软件测试过程的任务与管理也要符合软件 V&V 过程的有关规定。下面重点介绍软件 V&V 中的测试过程与管理。

（1）V&V 基本概念

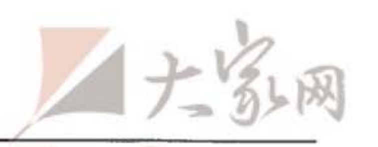
- 验证（Verification）：通过检查和提供客观证据，证实规定的需求已满足。
- 确认（Validation）：通过检查和提供客观证据，证实预期用途的需求是否得到满足。
- 独立验证和确认（IV&V Independent Verification and Validation）：由在技术、管理和财务上与开发组织有规定程度独立性的组织执行的 V&V 过程。

V&V 框架是由与软件开发过程同步的 V&V 过程、各阶段的 V&V 活动和任务组成的，如图 2-14 所示。

对每个 V&V 活动都规定了它的输入、任务和输出，如图 2-15 所示。

（2）软件 V&V 过程

- 软件生存周期的 V&V 过程框架。



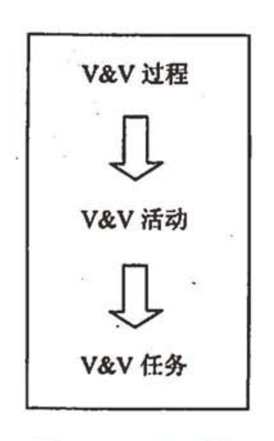


图 2-14　V&V 结构

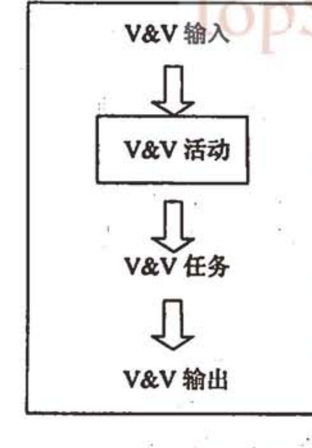


图 2-15　V&V 活动过程

整个软件生存周期的 V&V 过程框架结构描述了各阶段的 V&V 过程、活动和任务的层次关系，如图 2-16 所示。

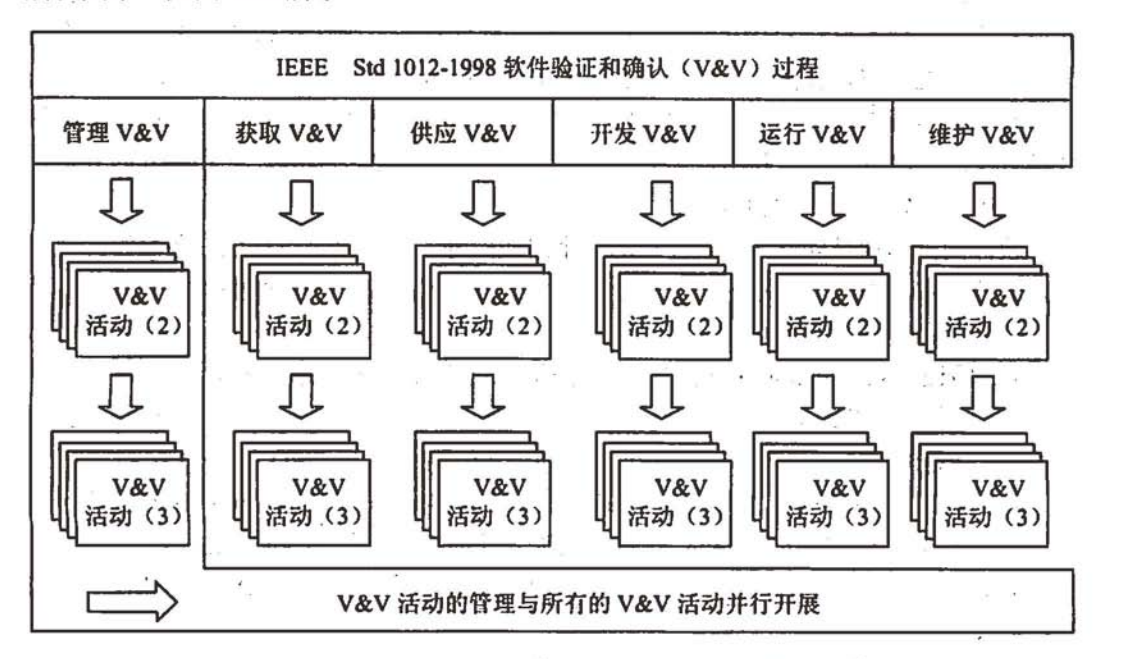


图 2-16　V&V 过程、活动和任务的层次关系

- 软件开发过程的 V&V 概述。

IEEE Std 1012-1998 中开发过程的 V&V，如图 2-17 所示。

软件开发过程

需　求	设　计	实　现	测　试
V&V 输入 • 系统需求 • 需求说明 • 接口规格说明 • 用户文档 • 系统测试计划 • 验收测试计划 • SVV 配置管理过程文档 • 危险分析报告 • 供方开发计划和进度安排 • V&V 任务结果	**V&V 输入** • 系统需求 • 需求说明 • 设计说明 • 接口规格说明 • 接口设计文档 • 设计标准 • 测试计划和设计 • 用户文档 • 危险分析报告 • 供方开发计划和进度安排 • V&V 任务结果	**V&V 输入** • 设计说明 • 接口设计文档 • 源代码 • 编码标准 • 用户文档 • 测试设计/用例 • 测试规程 • 单元测试结果 • 危险分析报告 • 供方开发计划和进度安排 • V&V 任务结果	**V&V 输入** • 测试计划、设计、用例和规程 • 设计说明 • 接口设计文档 • 源代码和可执行代码 • 用户文档 • 测试结果 • 危险分析报告 • 供方开发计划和进度安排 • V&V 任务结果
⇩	⇩	⇩	⇩
需　求	**设　计**	**实　现**	**测　试**
V&V 任务 • 可追踪性分析 • 软件需求评价 • 接口分析 • 系统 V&V 测试计划生成和验证 • 验收 V&V 测试计划生成和验证	**V&V 任务** • 可追踪性分析 • 软件设计评价 • 接口分析 • 部件 V&V 测试计划生成和验证 • 集成 V&V 测试计划生成和验证	**V&V 任务** • 可追踪性分析 • 源代码/源代码文档评价 • 接口分析 • V&V 测试用例生成和验证 • V&V 测试规程生成和验证	**V&V 任务** • 可追踪性分析 • 验收 V&V 测试规程生成和验证 • 集成 V&V 测试执行和验证 • 系统 V&V 测试执行和验证
⇩	⇩	⇩	⇩
V&V 输出 • 任务报告 • 异常报告 • 系统 V&V 测试计	**V&V 输出** • 任务报告 • 异常报告	**V&V 输出** • 任务报告 • 异常报告 • 部件/集成 V&V 测试用例 • 部件/集成/系统	**V&V 输出** • 任务报告 • 异常报告 • 验收 V&V 测试规程

图 2-17　软件开发过程的 V&V



(3) 软件 V&V 过程中的测试

- 测试过程。

GB/T 18905.5 中规定的开发过程中的软件测试过程包括：测试计划过程、测试设计过程、测试执行过程和测试结束过程。如图 2-18 所示。

项目管理　SOA 保证　配置管理

测试计划过程	测试设计过程	测试执行过程	测试结束过程
测试计划	测试说明	测试用例	测试结论
测试需求	测试方案	测试规程	测试报告

图 2-18　软件测试过程

- 需求 V&V 活动中的测试。

需求 V&V 活动中有两项测试任务：系统 V&V 测试计划生成和验证、验收 V&V 测试计划生成和验证。两项 V&V 的任务、输入和输出如表 2-6 所示。

表 2-6　需求 V&V 活动中的测试任务、输入和输出

V&V 任务		V&V 输入	V&V 输出
系统 V&V 测试计划生成和验证	• 策划系统 V&V 测试以确认软件需求 • 策划系统需求对测试设计、用例、规程和结果的追踪 • 验证系统 V&V 测试计划的制定是否遵循项目规定的测试文档目的格式和内容（参见 GB/T 9386） • 确认系统 V&V 测试计划满足下列准则： ① 系统需求的测试覆盖率 ② 采用测试方法和标准的符合性 ③ 对预期结果一致性 ④ 系统测试的可行性 ⑤ 执行和维护需求的可行性与可测性	• 系统需求 • 需求说明 • 接口规格说明 • 用户文档 • 系统测试计划	• 系统 V&V 测试计划 • 异常报告

续表

V&V 任务		V&V 输入	V&V 输出
验收 V&V 测试计划生成和验证	• 策划验收 V&V 测试以确认软件在运行环境下正确实现了系统和软件的需求 • 策划验收测试需求对测试设计、用例、规程和结果的追踪 • 验证验收 V&V 测试计划的制定是否遵循项目规定的测试文档目的格式和内容（参见 GB/T 9386） • 确认验收 V&V 测试计划满足下列准则： ① 系统需求的测试覆盖率 ② 对预期结果一致性 ③ 系统测试的可行性 ④ 执行和维护的可行性	• 系统需求 • 需求说明 • 接口规格说明 • 用户文档 • 验收测试计划	• 验收 V&V 测试计划 • 异常报告

- 设计 V&V 活动中的测试。

设计 V&V 活动中有三项测试任务：单元 V&V 测试计划生成和验证、集成 V&V 测试计划生成和验证与 V&V 测试计划生成和验证。三项 V&V 的任务、输入和输出如表 2-7 和表 2-8 所示。

表 2-7　设计 V&V 活动中的测试任务、输入和输出

V&V 任务		V&V 输入	V&V 输出
单元生成 V&V 测试计划生成和验证	• 策划单元 V&V 测试以确认单元正确实现了单元的要求。任务准则： ① 对设计要求的符合性 ② 功能和逻辑的符合性 ③ 模块的性能 ④ 边界、接口、出错 • 策划设计需求对测试设计、用例、规程和结果的追踪 • 验证单元 V&V 测试计划的制定是否遵循项目规定的测试文档目的格式和内容（参见 GB/T 9386） • 确认单元 V&V 测试计划满足下列准则： ① 对软件需求和设计的可追踪 ② 对软件需求和设计的外部一致性 ③ 单元需求件间的内部一致性	• 需求说明 • 设计说明 • 接口规格说明 • 接口设计文档 • 单元测试计划	• 单元 V&V 测试计划 • 异常报告



续表

V&V任务		V&V输入	V&V输出
单元生成V&V测试计划生成和验证	④ 每个单元的需求测试覆盖率 ⑤ 软件集成和测试的可行性 ⑥ 运行和维护的可行性		
集成V&V测试计划生成和验证	• 策划集成V&V测试以确认软件正确地实现了软件需求和设计。任务准则： ① 对每个集成阶段逐渐增大功能需求的一致性 ② 边界和强度条件下的性能 ③ 需求测试覆盖率 • 策划设计需求对测试设计、用例、规程和结果的追踪 • 验证集成V&V测试计划的制定是否遵循项目规定的测试文档目的格式和内容（参见GB/T 9386） • 确认集成V&V测试计划满足下列准则： ① 系统需求的可追踪性 ② 系统需求的外部和内部一致性 ③ 软件的需求测试覆盖率 ④ 测试标准和方法的合适性 ⑤ 软件合格性测试的可能性 ⑥ 运行和维护的可行性	• 需求说明 • 设计说明 • 接口规格说明 • 接口设计文档 • 集成测试计划	• 集成V&V测试计划 • 异常报告

表 2-8 设计V&V活动中的测试任务、输入和输出(续)

V&V任务		V&V输入	V&V输出
V&V测试设计生成和验证	• 设计如下测试： ① 单元测试 ② 集成测试 ③ 系统测试 ④ 验收测试 • 一系列V&V测试计划要求的追踪 • 验证V&V测试计划的制定是否遵循项目规定的测试文档目的格式和内容(参见GB/T 9386) • 确认V&V测试设计满足V&V的下列任务：	• 需求说明 • 接口规格说明 • 设计说明 • 用户文档 • 测试计划 • 测试设计	• 单元V&V测试计划 • 集成V&V测试计划 • 系统V&V测试计划 • 验收V&V测试计划 • 异常报告

续表

V&V 任务		V&V 输入	V&V 输出
V&V 测试设计生成和验证	① 单元 V&V 测试计划生成和验证 ② 集成 V&V 测试计划生成和验证 ③ 系统 V&V 测试计划生成和验证 ④ 验收 V&V 测试计划生成和验证相关测试的各自的准则		

- 实现 V&V 活动中的测试。

实现 V&V 活动中有三项测试任务：V&V 测试用例生成和验证、V&V 测试规程生成和验证以及部件 V&V 测试计划执行和验证。三项 V&V 的任务、输入和输出如表 2-9 所示。

表 2-9　实现 V&V 活动中的测试任务、输入和输出

V&V 任务		V&V 输入	V&V 输出
V&V 测试用例生成和验证	• 开发如下测试用例： ① 单元测试 ② 集成测试 ③ 系统测试 ④ 验收测试 • 一系列 V&V 测试用例要求的追踪 • 验证 V&V 测试用例的制定是否遵循项目规定的测试文档目的格式和内容（例如 GB/T 9386） • 确认 V&V 测试设计满足 V&V 的下列任务： ① 单元 V&V 测试计划生成和验证 ② 集成 V&V 测试计划生成和验证 ③ 系统 V&V 测试计划生成和验证 ④ 验收 V&V 测试计划生成和验证相关测试的各自的准则	• 需求说明 • 接口规格说明 • 设计说明 • 接口设计文档 • 用户文档 • 测试设计 • 测试用例	• 单元 V&V 测试用例 • 集成 V&V 测试用例 • 系统 V&V 测试用例 • 验收 V&V 测试用例 • 异常报告
V&V 测试规程生成和验证	• 开发如下测试规程： ① 单元测试 ② 集成测试 ③ 系统测试	• 需求说明 • 接口规格说明 • 设计说明 • 接口设计文档	• 单元 V&V 测试规程 • 集成 V&V 测试规程 • 系统 V&V 测试规程 • 异常报告

续表

	V&V 任务	V&V 输入	V&V 输出
V&V 测试规程生成和验证	④ 验收测试 • 一系列 V&V 测试规程要求的追踪 • 验证 V&V 测试规程的制定是否遵循项目规定的测试文档目的格式和内容（参见 GB/T 9386） • 确认 V&V 测试规程满足 V&V 的下列任务： ① 单元 V&V 测试计划生成和验证 ② 集成 V&V 测试计划生成和验证 ③ 系统 V&V 测试计划生成和验证 ④ 验收 V&V 测试计划生成和验证相关测试的各自的准则	• 用户文档 • 测试用例 • 测试规程	
单元 V&V 测试执行和验证	• 执行单元 V&V 测试 • 分析测试结果以确认软件正确实现了设计 • 确认测试结果可追踪到在测试计划文档中规定的测试准则 • 按单元 V&V 测试计划的要求记录结果 • 使用单元 V&V 测试结果确认软件满足了 V&V 测试验收准则 • 记录实际的和预期的测试结果间的差异	• 源代码 • 可执行代码 • 设计说明 • 接口设计文档 • 单元测试计划 • 单元测试规程 • 单元测试结果	• 对任务报告测试结果 • 异常报告

（4）软件测试 V&V 活动

测试 V&V 活动覆盖了集成测试、系统测试和验收测试。测试 V&V 活动及它与软件生存期的关系如图 2-19 所示。V&V 的目标是确保通过执行集成测试、系统测试和验收测试使软件需求和分配给软件的系统需求得到满足。

测试的 V&V 工作应生成自己的 V&V 测试件（包括计划、设计、用例和规程），执行并记录自己的测试，并对照软件需求验证测试计划、设计、用例、规程和结果；测试的 V&V 工作应验证测试活动和测试件（包括计划、设计、用例、规程和执行结果）。测试 V&V 活动的任务、输入与输出的关系如表 2-10 和表 2-11 所示。

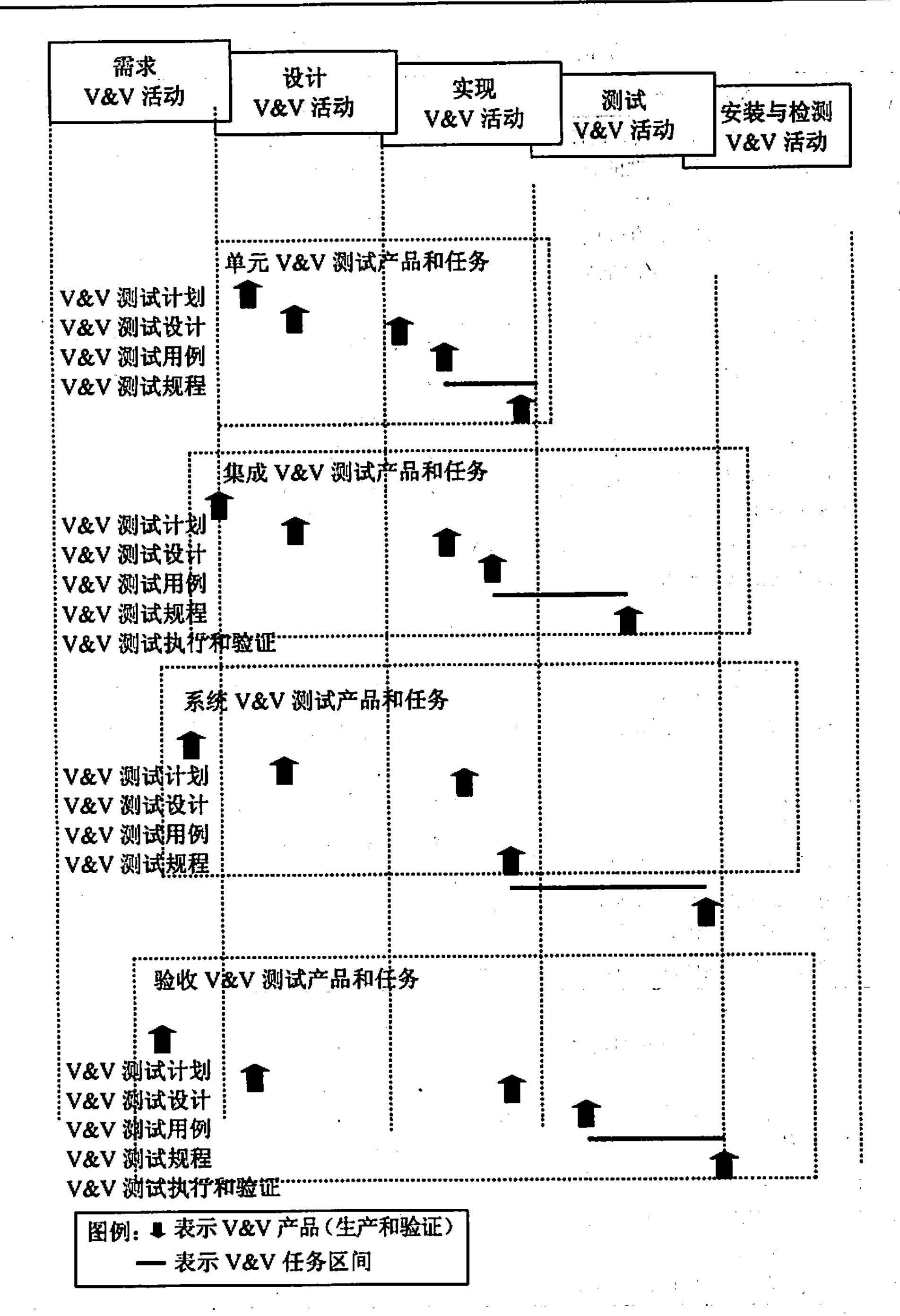


图 2-19　V&V 测试产品和测试执行任务的时段图



表 2-10 测试 V&V 活动中的测试任务、输入和输出

	V&V 任务	V&V 输入	V&V 输出
可追踪性分析	• 分析在 V&V 测试计划、设计、用例、规程中的正确性和完备性的关系 • 对于正确性，验证 V&V 测试计划、设计、用例、规程间关系的有效性 • 对于完备性，验证所有 V&V 测试规程可追踪到测试计划	• V&V 测试计划 • V&V 测试设计 • V&V 测试规程	• 任务报告——可追踪性分析报告 • 异常报告
验收 V&V 测试规程生成和验证	• 制定验收 V&V 测试规程 • 一系列关于验收 V&V 测试计划要求的追踪 • 验证验收 V&V 测试规程的制定是否遵循项目规定的测试文档目的格式和内容（参见 GB/T 9386） • 确认验收 V&V 测试规程满足验收 V&V 测试计划生成和验收任务的准则	• 设计说明 • 接口规格文档 • 用户文档 • 验收测试计划 • 验收测试规程	• 验收 V&V 测试规程 • 异常报告
集成 V&V 测试执行和验证	• 执行集成 V&V 测试 • 分析测试结果以验证软件单元是否正确集成 • 确认测试结果可追踪到在测试计划文档中规定的测试准则 • 按集成 V&V 测试计划的要求记录结果 • 使用集成 V&V 测试结果确认软件是否满足 V&V 测试验收准则 • 记录实际的和预期的测试结果间的差异	• 源代码 • 可执行代码 • 验收测试计划 • 集成测试规程 • 集成测试结果	• 任务报告——测试结果 • 异常报告
系统 V&V 测试执行和验证	• 执行系统 V&V 测试 • 分析测试结果以确认软件是否满足系统需求 • 确认测试结果可追踪到在测试计划文档中规定的测试准则 • 按系统 V&V 测试计划的要求记录结果 • 使用系统 V&V 测试结果确认软件是否满足 V&V 测试验收准则 • 记录实际的和预期的测试结果间的差异	• 源代码 • 可执行代码 • 系统测试计划 • 系统测试规程 • 系统测试结果	• 任务报告——测试结果 • 异常报告

表 2-11　测试 V&V 活动中的测试任务、输入和输出（续）

	V&V 任务	V&V 输入	V&V 输出
验收 V&V 测试执行生成和验证	• 执行验收 V&V 测试 • 分析测试结果以确认软件是否满足系统需求 • 确认测试结果可追踪到在测试计划文档中规定的测试准则 • 按验收 V&V 测试计划的要求记录结果 • 使用验收 V&V 测试结果确认软件是否满足 V&V 测试验收准则 • 记录实际的和预期的测试结果间的差异	• 源代码 • 可执行代码 • 用户文档 • 验收测试计划 • 验收测试规程 • 验收测试结果	• 任务报告——测试结果 • 异常报告
危险分析	验证测试设施没有引进新的风险，更新危险分析	• 源代码 • 可执行代码 • 测试结果 • 危险分析报告	• 任务报告——危险分析报告 • 异常报告
风险分析	使用先前的任务报告评审和更新风险分析为消除、降低和缓解风险提供建议	• 供方开发计划和进度安排 • 危险分析报告 • V&V 任务结果	• 任务报告——风险分析报告 • 异常报告

2.8　软件失效分类与管理

2.8.1　软件失效分类

软件测试使用各种术语描述软件出现的问题，通用的术语如下：

- 软件错误（software error）。
- 软件缺陷（software defect）。
- 软件故障（software fault）。
- 软件失效（software failure）。

区分这些术语的概念很重要，它关系到测试工程师对软件失效现象与机理的深刻理解，而这些概念常常在文献中被混淆。本书将根据国际经典软件测试论著，对软件失效的机理进行阐述。

由于软件内部逻辑复杂，运行环境动态变化，且不同的软件差异可能很大，因而软件失效机理可能有不同的表现形式。但总的说来，软件失效机理可描述为：软件错误→软件缺陷→软件故障→软件失效。

① 软件错误：在可以预见的时期内，软件仍将由人来开发。在整个软件生存期的各

个阶段，都贯穿着人的直接或间接的干预。然而，人难免犯错误，这必然给软件留下不良的痕迹。软件错误是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。可见，软件错误是一种人为过程，相对于软件本身，是一种外部行为。

② 软件缺陷：软件缺陷是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差，如少一逗点、多一语句等。其结果是软件运行于某一特定条件时出现软件故障，这时称软件缺陷被激活。

③ 软件故障：软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态。譬如，软件处于执行一个多余循环过程时，我们说软件出现故障。此时若无适当措施（容错）加以及时处理，便产生软件失效。显然，软件故障是一种动态行为。

④ 软件失效：软件失效是指软件运行时产生的一种不希望或不可接受的外部行为结果。

综上所述，软件错误是一种人为错误。一个软件错误必定产生一个或多个软件缺陷。当一个软件缺陷被激活时，便产生一个软件故障；同一个软件缺陷在不同条件下被激活，可能产生不同的软件故障。软件故障如果没有及时的容错措施加以处理，便不可避免地导致软件失效；同一个软件故障在不同条件下可能产生不同的软件失效。

在软件生存期中存在和产生形形色色的软件错误、缺陷、故障和失效。不同的软件，其错误、缺陷、故障和失效无论在表现形式、性质乃至数量上都可能大不相同，试图对它们作一个全面而详细的阐述是不现实的，所以有必要加以区别对待。关于“错误”的广义定义是：不正确的事务和行为。在 1999 年（美）John D. Musa 的《软件可靠性工程》书中，关于“软件错误”是这样描述的：“错误是在系统运行时，引起或可能潜在地引起失效的缺陷，是一种面向开发的概念。”例如，当用户单击某个具体的菜单时，本应在屏幕上出现特定的对话框，但是却没有出现。这种行为就是一个失效。造成这种失效的错误可能是遗漏代码。这里给出的定义是“电气与电子工程师协会（IEEE）”和“美国标准协会（ASA）”的标准，是通过引起失效和错误的系统成分，来定义失效和错误的。这些成分一般是硬件、软件和人。

John D. Musa 在 1999 年对软件错误的定义是：软件错误是代码中的缺陷，是由错误引起的，是由一个或多个人的不正确或遗漏行为造成的。例如，系统工程师在定义需求时可能会犯错误，从而导致代码错误，而代码错误又导致在一定条件下执行系统时出现失效。“缺陷”是指欠缺或不够完备的地方。软件的欠缺和不完备主要是针对产品说明书而言的。2001 年（美）Ron Patten 著的《软件测试》一书对软件缺陷进行了定义。按照一般定义，只要软件出现的问题符合下列 5 种情况的任何一种，就叫做软件缺陷：① 软件未达到产品说明书中标明的功能；② 软件出现了产品说明书中指明的不会出现的错误；③ 软件功能超出了产品说明书指明的范围；④ 软件未达到产品说明书虽未指出但

应达到的目标；⑤ 软件测试人员认为软件难以理解、不易使用、运行速度慢，或最终用户认为不好使用。实践表明，大多数软件缺陷产生的原因并非源自编程错误，主要来自于产品说明书的编写和产品方案设计。

产品说明书成为软件缺陷的罪魁祸首，是因为产品说明书编写得不全面、不完整和不准确，而且经常更改，或者整个开发组没有很好地沟通和理解。这也就是出自于软件需求说明书本身的问题，或开发人员对需求说明书的理解与沟通不足。

软件缺陷的第二大来源是设计方案，也就是软件设计说明书。这是程序员开展软件计划和构架的地方，就像建筑师为建筑物绘制蓝图一样。这里产生软件缺陷的原因与产品说明书或需求说明书是一样的，片面、多变、理解与沟通不足。

总之，软件缺陷是开发的软件与软件需求说明书、设计说明书的不一致；软件的实现未满足应达到目标的用户潜在需求。故障是指一个实体发生障碍和毛病。软件故障在 ISO 14598 软件产品评价标准中的定义是：计算机程序中的不正确的步骤、过程或数据定义。

软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态，软件出现故障若无适当措施（容错）加以及时处理，便产生软件失效。显然，软件故障是一种动态行为。

在软件设计和编程过程中，花费很大的精力确保软件系统能从各种故障导致的失效中恢复。当遇到软件出现故障时，系统不能像软件设计和用户要求那样运行而导致失效，就需要有故障恢复措施，以保证故障恢复后的继续执行。软件失效是系统行为对用户要求的偏离，是一种面向用户的概念。这就是说，失效意味着系统的运行。只有在执行程序过程中才会发现软件失效，发现的潜在失效可以是设计审查、代码阅读和其他方法产生的结果。有的项目组还把文档错误计算在软件错误之内，这一般是不正确的，因为文档并不直接影响程序的执行。因为用户接受的是程序使用的错误信息，文档错误可能会导致用户的失效。但是，用户并不是软件成分，不能把用户看成是与失效和可靠性有关的单独的系统成分。

对失效严重程度进行分类，主要是为了结合失效频率来解决失效优先级的确定。常见的分类标准包括对人员生命、成本和系统能力的影响。失效强度常常应用于软件可靠性工程中，最初是指单位时间内的失效次数；基于软件大量的使用经验，失效强度表示为每个自然单元出现的失效数目更加方便。失效强度是表示可靠性的另一种形式。

关于概念不可能彼此分得很清楚，实际上也没有太大的必要。目前软件测试界一般主要使用缺陷（defect）和错误（error）这两个词。在测试过程中，我们找到的错误会有不同的类型，对错误的分析与管理是十分重要的。

2.8.2　缺陷与错误分布

通过对错误分布情况的仔细分析，可以帮助我们将测试的主要精力更好地集中到最



有价值的地方，如图 2-20 显示了缺陷与错误的分布情况。

开发早期的错误通常是很多的，而且令人讨厌的是它们还会转移到后期。这和制造业的装配线类似，如果一个坏零件或次品被允许上线，从这点开始，包含它的组件就是“坏”的，如果该组件下了线，并出了厂门，情况就会更糟糕，就得为那个坏零件付出代价。换句话说，错误不是自封闭的，当它们转移到后面的组件中时，往往会以新的形势出现。

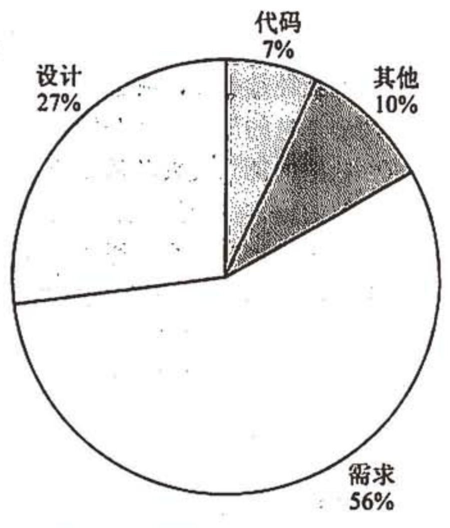


图 2-20 缺陷与错误的分布情况

所有的错误都是要付出代价的。没有被发现的错误，以及那些在开发过程中很晚才被发现的错误都是成本非常高的，没有被发现的错误就在系统中迁移、扩散，最终导致系统失效。直到很晚才被发现的错误常常造成代价昂贵的返工。那些没有被发现的错误导致系统失效，造成严重的财产损失，有时还会带来法律上的麻烦，系统将终身为此付出高昂的代价。

这意味着测试是贯穿开发全过程的工作，也意味着对最终产品的测试仅仅是软件质量大战中的一个战役，而且不是代价最高的战役。今天，40%～70%的软件开发时间和资源都花在查错和纠错上。不幸的是大多数组织还没有一套办法来准确地计算成本，不管怎样，在使用测试资源方面任何有意义的改进都能极大地降低开发成本。

2.8.3 缺陷与错误严重性和优先级

软件存在的缺陷与错误会带来软件失败的风险，重要软件故障与失效会导致重大经济损失与灾难。在报告软件缺陷时，一般要讲明如何处置它们。测试员要对软件缺陷分类，以简明扼要的方式指出其影响，以及修改的优先次序。给软件缺陷与错误划分严重性和优先级的通用原则是：

① 表示软件缺陷所造成的危害的恶劣程度；

② 优先级表示修复缺陷的重要程度与次序。

- 严重级。

① 严重：系统崩溃、数据丢失、数据毁坏。

② 较严重：操作性错误、错误结果、遗漏功能。

③ 一般：小问题、错别字、UI 布局、罕见故障。

④ 建议：不影响使用的瑕疵或更好的实现。

- 优先级。

① 最高优先级：立即修复，停止进一步测试。

② 次高优先级：在产品发布之前必须修复。

③ 中等优先级：如果时间允许应该修复。

④ 最低等优先级：可能会修复，但是也能发布。

一般的严重性和优先级的划分用数字 1～4 表示，有的小数字表示的级别最高；而有的用大数字表示级别高。同样的错误和缺陷，在不同的开发过程或软件的不同部分，严重性和优先级将有所变化，要具体情况具体分析。

2.8.4　软件错误跟踪管理

软件测试的主要目的在于发现软件存在的错误（Bug），如何处理测试中发现的错误，将直接影响到测试的效果。只有正确、迅速、准确地处理这些错误，才能消除软件错误，保证要发布的软件符合需求设计的目标。在实际的软件测试过程中，每个 Bug 都要经过测试、确认、修复、验证等的管理过程，这是软件测试的重要环节。

1．错误跟踪管理

为了正确地跟踪每个软件错误的处理过程，通常将软件测试发现的每个错误作为一条记录输入指定的错误跟踪管理系统。

目前已有的错误跟踪管理软件包括 Compuware 公司的 TrackRecord 软件（商业软件）、Mozilla 公司的 Buzilla 软件（免费软件），以及国内的微创公司的 BMS 软件，这些软件在功能上各有特点，可以根据实际情况选用。当然，也可以自己开发缺陷跟踪软件，例如基于 Notes 或是 ClearQuese 开发的错误跟踪管理软件。

作为一个错误跟踪管理系统，需要正确记录错误信息和错误处理信息的全部内容。

（1）Bug 记录信息

主要包括以下几项内容。

- 测试软件名称；
- 测试版本号；
- 测试人名称；
- 测试事件；
- 测试软件和硬件配置环境；
- 发现软件错误的类型；
- 错误的严重等级；
- 详细步骤；
- 必要的附图；



- 测试注释。

（2）Bug 处理信息

主要包括以下 4 项内容。

- 处理者姓名；
- 处理时间；
- 处理步骤；
- 错误记录的当前状态。

正确的错误数据库权限管理是错误跟踪管理系统的重要考虑要素，一般要保证对于添加的错误不能从数据库中删除。

2．软件错误的状态

软件错误的主要状态包括以下内容。

- 新信息（New）：测试中新报告的软件 Bug。
- 打开（Open）：被确认并分配给相关开发人员处理。
- 修正（Fixed）：开发人员已完成修正，等待测试人员验证。
- 拒绝（Declined）：拒绝修改 Bug。
- 延期（Deferred）：不在当前版本修复的错误，下一版修复。
- 关闭（Closed）：Bug 已被修复。

3．错误管理流程

错误管理的流程可以概括为以下几项内容。

- 测试人员提交新的错误入库，错误状态为“New”。
- 高级测试人员验证错误。

① 如果确认是错误，分配给相应的开发人员，设置状态为“Open”。

② 如果不是错误，则拒绝，设置为“Declined”状态。

- 开发人员查询状态为“Open”的错误，做如下处理。

① 如果不是错误，则置状态为“Declined”。

② 如果是错误，则修复并置状态为“Fixed”。

③ 如果不能解决的错误，要留下文字说明并保持错误为“Open”状态。

④ 对于不能解决和延期解决的错误，不能由开发人员自己决定，一般要通过某种会议（评审会）通过才能认可。

- 测试人员查询状态为“Fixed”的错误，验证错误是否已解决，做如下处理。

① 如问题解决了，置错误的状态为“Closed”。

② 如问题没有解决，则置状态为“Reopen”。

4．错误流程管理原则

错误流程管理遵照以下原则。

① 为了保证错误处理的正确性，需要有丰富测试经验的测试人员验证发现的错误是否是真正的错误，书写的测试步骤是否准确，可以重复。

② 每次对错误的处理都要保留处理信息，包括处理姓名、时间、处理方法、处理意见、Bug 状态。

③ 拒绝或延期处理错误不能由程序员单方面决定，应该由项目经理、测试经理和设计经理共同决定。

④ 错误修复后必须由报告错误的测试人员验证，确认已经修复后，才能关闭错误。

- 加强测试人员与程序员之间的交流，对于某些不能重复的错误，可以请测试人员补充详细的测试步骤和方法，以及必要的测试用例。

2.9　白盒测试

白盒测试也称结构测试或逻辑驱动测试，它是按照程序内部的结构测试程序，通过测试来检测产品内部动作是否按照设计规格说明书的规定正常进行，检验程序中的每条通路是否都能按预定要求正确工作。

这一方法是把测试对象看作一个打开的盒子，测试人员依据程序内部逻辑结构相关信息，设计或选择测试用例，对程序所有逻辑路径进行测试，通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致。

采用什么方法对软件进行测试呢？常用的软件测试方法有两大类：静态测试方法和动态测试方法。其中软件的静态测试不要求在计算机上实际执行所测程序，主要以一些人工的模拟技术对软件进行分析和测试；而软件的动态测试是通过输入一组预先按照一定的测试准则构造的实例数据来动态运行程序，而达到发现程序错误的过程。这两种测试方法和相应测试用例的设计方法将在相应章节作详细介绍。

2.10　黑盒测试

黑盒测试也称功能测试，它是通过测试来检测每个功能是否都能正常使用。在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。黑盒测试着眼于程序外部结构，不考虑内部逻辑结构，主要针对软件界面和软件功能进行测试。



黑盒测试是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的。很明显，如果外部特性本身有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。

黑盒测试法注重于测试软件的功能需求，主要试图发现下列几类错误。

- 功能不正确或遗漏；
- 界面错误；
- 数据库访问错误；
- 性能错误；
- 初始化和终止错误等。

从理论上讲，黑盒测试只有采用穷举输入测试，把所有可能的输入都作为测试情况考虑，才能查出程序中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但可能的输入进行测试。这样看来，完全测试是不可能的，所以我们要进行有针对性的测试，通过制定测试案例指导测试的实施，保证软件测试有组织、按步骤，以及有计划地进行。黑盒测试行为必须能够加以量化，才能真正保证软件质量，而测试用例就是将测试行为具体量化的方法之一。具体的黑盒测试用例设计方法包括等价类划分法、边界值分析法、错误推测法、因果图法、判定表驱动法、正交试验设计法、功能图法等。

等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值。

边界值分析是通过选择等价类边界的测试用例。边界值分析法不仅重视输入条件边界，而且也必须考虑输出域边界。

错误推测设计方法就是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。

因果图方法是从用自然语言书写的程序规格说明的描述中找出因（输入条件）和果（输出或程序状态的改变），可以通过因果图转换为判定表。

正交试验设计法，就是使用已经造好了的正交表格来安排试验并进行数据分析的一种方法，目的是用最少的测试用例达到最高的测试覆盖率。

2.11 自动化测试

2.11.1 自动化测试的基本概念

1. 自动化测试的引入

软件测试作为保证软件质量和可靠性的关键技术，正日益受到广泛的重视，但随着

软件工程的规模越来越大，客户对软件的质量要求越来越高，测试的工作量也越来越大。如何进行测试，如何提高测试的质量和效率，从而确保软件产品的质量和可靠性，就成了许多人深感困扰的问题。

目前，企业级应用系统越来越多，这些系统可能包括 ERP 系统，CRM 系统等。这些系统在发布之前或升级之后都要经过测试，确保主要功能都能正常运行，错误最少。如何有效地测试不断升级和不断更换应用环境的应用系统，是每个公司都会面临的问题。如果时间或资源有限，这个问题会更加棘手。人工测试的工作量太大，同时还需要额外的时间来培训测试人员等。为了确保那些复杂的企业级应用在不同环境下都能可靠地运行，需要一个能简单操作的测试工具来自动完成应用程序的功能性测试。

同时，目前企业的网络应用环境都必须支持大量用户和不同的软硬件应用环境。难以预知的用户负载和越来越复杂的应用环境使公司时时担心会发生用户响应速度过慢、系统崩溃等问题。这些都不可避免地导致公司收益的损失。为了在终端用户正式使用前，对应用系统各个环节的质量、可靠性和可扩展性进行测试和评价，就需要适用于不同体系架构的自动负载压力测试工具，以预测系统行为并为系统优化提供依据。

总之，为了更加快速、有效地对软件进行测试，提高软件产品的质量，我们必然会利用测试工具，也必然会引入自动化测试。

2．自动化测试的定义

自动化测试就是通过测试工具或其他手段，按照测试工程师的预定计划对软件产品进行自动的测试，它是软件测试的一个重要的组成部分，它能够完成许多手工无法完成或者难以实现的一些测试工作。正确、合理地实施自动化测试，能够快速、全面地对软件进行测试，从而提高软件质量，节省经费，缩短产品发布周期。

软件测试自动化涉及到测试流程、测试体系、自动化编译以及自动化测试等方面的整合。也就是说，要让测试能够自动化，不仅是技术、工具的问题，更是一个公司和组织的文化问题。首先公司要从资金、管理上给予支持，其次要有专门的测试团队去建立适合自动化测试的测试流程和测试体系；最后才是把源代码从受控库中取出、编译、集成、发布并进行自动化的功能和性能等方面的测试。

2.11.2　自动化测试的优势与局限

1．自动化测试的优势

自动化测试能够替代大量手工测试工作，避免重复测试，同时，它还能够完成大量手工无法完成的测试工作，如并发用户测试、大数据量测试、长时间运行可靠性测试等，概括地说，自动化测试具有如下优点。

- 提高测试质量：软件开发的过程就是一个持续集成和改进的过程，而每一次修



改都有可能产生错误。因此，当软件产品的一部分，或者全部，或者应用环境被修改时都需要对软件产品重新进行测试，其目的是验证修改后的系统或者产品的质量是否符合规格说明。例如，对于产品型的软件，每发布一个新的版本，其中大部分功能和界面都和上一个版本相似或完全相同，这部分功能特别适合于自动化测试，由于自动测试工具提供了简便的回归测试，能以便利的方式验证是否有新的错误进入软件产品，既节省了重复手工输入的工作量，保证了测试案例的一致性，避免了人为因素，也可以使测试达到测试每个质量特性的目的，从而提高软件测试的质量。

- 提高测试效率，缩短测试工作时间：软件系统的规模越来越大，功能点越来越多，达到几千个上万个，人工测试非常耗时和繁琐，这样必然会导致测试效率低下，而自动化测试工具可以较好地执行这些频繁的测试任务。在充分并合理地使用了测试工具以后，可以减轻测试工程师的手工测试工作，同时，测试工具还可以把控制和管理引入整个测试过程，能够保证测试的进度。
- 提高测试覆盖率：通过自动化测试工具的录制回放及数据驱动来测试功能，可以提高测试覆盖率。通过测试工具的辅助分析功能，可以提高测试的深度。
- 执行手工测试不能完成的测试任务：有些非功能性方面的测试，例如，压力测试、负载测试、大数据量测试、崩溃性测试等，人工测试是不可能实现的。例如，找若干台电脑和同样数目的操作人员在同一时刻进行操作，然后拿秒表记录下反应时间，这样的手工作坊式的测试方法不切实际且无法捕捉程序内部变化情况。
- 更好地重现软件缺陷的能力：自动化测试具有更好的一致性和可重复性，由于每次自动化测试运行的脚本是相同的，所以每次执行的测试具有一致性，人是很难做到的。由于自动化测试的一致性，很容易发现被测软件的任何改变。
- 更好地利用资源：理想的自动化测试能够按计划完全自动地运行，在开发人员和测试人员不可能实行三班倒的情况下，自动化测试可以胜任这个任务，例如，完全可以在周末或者晚上执行测试。这样充分地利用资源，也避免了开发和测试之间的冲突。
- 增进测试人员与开发人员之间的合作伙伴关系：测试工程师为了更好地使用自动化测试工具，需要对开发技术有深入的理解和实践，因此测试工程师也有了与开发工程师更多、更平等地交流的机会，自动化测试为测试工程师与程序开发人员协同工作提供了一种便利的手段，双方将有更多的合作与尊重。

测试工具能够提高软件质量，改进测试过程，因此在许多公司中得到了广泛应用，由于自动化测试工具自身的特点，为达到较高的投资回报率，在以下项目和环境中更适

合使用自动化测试工具。

- 需要反复进行的工作。在持续修改软件功能的项目中，对功能的测试需要反复进行，人工测试工作量极大。功能性测试工具能够自动进行重复性的工作，验证软件的修改是否引入了新的缺陷，旧的缺陷是否已经修改。减少人工测试的工作量。
- 负载压力测试。负载压力测试需要模拟大量并发用户和大数据量，这样的测试用手工不能完成或不能很好地完成，而自动化测试工具则可以很好地解决这个问题，在测试脚本运行过程中也不需要人工干预，能够充分利用非工作时间。
- 公司有大量的测试人员和开发人员，他们合作完成一个产品，那么如何在产品的生命周期中进行有效管理和合作，借助于自动化的测试管理工具，会取得事半功倍的效果。
- 如果需要进行测试系统后台或者内部的性能特性，进而进行故障定位和性能调优，自动化测试工具会是一个不错的选择。

2．自动化测试的局限性

虽然自动化测试可以提高测试效率，能够完成手工测试不能完成的工作，但自动化测试在实际应用中也存在局限性，并不能完全替代手工测试，在下面的领域中自动化测试会有一定的局限性。

- 定制型项目：为客户定制的项目，甚至采用的开发语言、运行环境也是客户特别要求的，开发公司在这方面的测试积累少，这样的项目不适合作自动化功能测试。
- 周期很短的项目：项目周期很短，相应的测试周期也很短，因此花大量精力准备的测试脚本，不能得到重复地利用。当然，为了某种特定的测试目的专门执行的测试任务除外，比如，针对特定应用的性能测试等。
- 业务规则复杂的对象：业务规则复杂的对象有复杂的逻辑关系和运算关系，工具很难实现，或者要实现这些测试过程，需要投入的测试准备时间比直接进行手工测试所需的时间更长。
- 人体感观与易用性测试：界面的美观、声音的体验、易用性的测试，无法用测试工具来实现。
- 不稳定的软件：如果软件不稳定，则会由于这些不稳定因素导致自动化测试失败，或者致使测试结果本身就是无效的。
- 涉及物理交互：自动化测试工具不能很好地完成与物理设备的交互，比如刷卡器的测试等。



任何工具都有它的可用范围，就像我们不能拿剪刀去劈柴，不能拿斧头去裁减布料一样，面对任何一个待测系统，我们也应该考虑选用的测试工具是否合适，引入测试工具是否有利于该项目的开发等，否则，有可能适得其反。

以上介绍了自动化测试的局限性，因此，作为测试工程师，在考虑选用自动化测试的过程中，还需要了解公司领导、项目负责人等对于自动化测试的期望并消除他们一些不正确的期望，如下所示。

- 自动化测试可以完成一切测试工作：很多人一听到测试自动化，就认为自动化测试工具可以完成一切测试工作，从测试计划到测试执行，再到测试结果分析，不需要任何人工干预等，很显然，这是一种理想状态，现实中还没有哪个测试工具有这个能力，并且将来也不会有。在现实中有关的测试设计、测试案例以及一些关键的测试任务还是需要人工参与的，即自动化测试是对手工测试的辅助和补充，它永远不可能取代手工测试。
- 测试工具可适于所有的测试：每种自动化测试工具都有它的应用范围和可用对象，所以不能认为一种自动化测试工具能够满足所有的测试需求。针对不同的测试目的和测试对象，我们应该选择合适的测试工具来对它进行测试，在很多情况下，需要利用多种测试工具才能完成测试工作。
- 测试工具能使工作量大幅度降低：事实上，引入自动化测试工具不会马上减轻测试工作，相反，在更多情况下，首次将自动化测试工具引入企业时，测试工作实际上变得更艰巨了。只有在正确合理地使用测试工具，并有一定的技术积累后，测试工作量才能逐渐减轻。
- 测试工具能实现百分之百的测试覆盖率：由于自动化测试可以增加测试覆盖的深度和广度，比如，利用白盒测试工具可能实现语句全覆盖、逻辑路径全覆盖等，但因为穷举测试必须使用所有可能的数据，包括有效的和无效的测试数据，所以在有限的资源下也不可能进行百分之百的彻底测试。
- 自动化测试工具容易使用：对于这一点，很多测试工程师也有同样的错误观点，认为测试工具可以简单地通过捕获（录制）客户端操作生成脚本，且脚本不加编辑就可用于回放使用。事实上，自动化测试不是那么简单，捕获的操作是否正确以及脚本编辑是否合理都会影响测试结果，因此，自动化测试需要更多的技能，也需要更多的培训。
- 自动化测试能发现大量的新缺陷：发现更多的新缺陷应该是手工测试的主要目的，不能期望自动化测试去发现更多新缺陷，事实上自动化测试主要用于发现老缺陷。

2.11.3 选择合适的自动化测试工具

1. 自动化测试工具分类

自动化测试工具可以减少测试工作量，提高测试工作效率，但首先是能够选择一个合适的且满足企业信息系统工程环境的自动化测试工具，因为不同的测试工具，其面向的测试对象是不一样的。按照测试工具的主要用途和应用领域，可以将自动化测试工具分为以下几类。

- 负载压力测试工具：这类测试工具的主要目的都是为了度量应用系统的可扩展性和性能，是一种预测系统行为和性能的自动化测试工具。它们通过模拟成百上千直至上万用户并发执行关键业务，而完成对应用程序的测试，在实施并发负载过程中通过实时性能监测来确认和查找问题，并针对所发现问题对系统性能进行优化，确保应用的成功部署。负载压力测试工具能够对整个企业架构进行测试，通过这些测试，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。这类工具的主要代表有 LoadRunner、QALoad、SILK PERFORMA V 和 E-Test Suite 等。
- 功能测试工具：通过自动录制、检测和回放用户的应用操作，将被测系统的输出记录同预先给定的标准结果进行比较，功能测试工具能够有效地帮助测试人员对复杂的企业级应用的不同发布版本的功能进行测试，提高测试人员的工作效率和质量。其主要目的是用于检测应用程序是否能够达到预期的功能并正常运行。功能测试工具可以大大减轻黑盒测试的工作量，在迭代开发的过程中，能够很好地进行回归测试。这类工具的主要代表有 WinRunner、QARun 等。
- 白盒测试工具：白盒测试工具一般是针对代码进行测试，测试中发现的缺陷可以定位到代码级，根据测试工具原理的不同，又可以分为静态测试工具和动态测试工具。静态测试工具直接对代码进行分析，不需要运行代码，也不需要对代码编译链接和生成可执行文件。静态测试工具一般是对代码进行语法扫描，找出不符合编码规范的地方，根据某种质量模型评价代码的质量，生成系统的调用关系图等。静态测试工具的代表有 Logiscope 软件和 PRQA 软件。动态测试工具与静态测试工具不同，动态测试工具一般采用“插桩”的方式，向代码生成的可执行文件中插入一些监测代码，用来统计程序运行时的数据。其与静态测试工具最大的不同就是动态测试工具要求被测系统实际运行。动态测试工具的代表有 DevPartner、Rational Purify 系列等。
- 网络测试工具：这类工具主要包括网络故障定位工具、网络性能监测工具、网络仿真模拟工具等。它们分析分布式应用性能，关注应用、网络和其他元素（如



服务器）内部的交互式活动，以便使网络管理员能够了解网络不同位置和不同活动之间应用的行为。你可以用它在交易执行过程中、Web 查找和检索中或在日常数据库上载/下载中跟踪应用行为。它可在会话级、代码级，甚至在帧级和包级观察应用的行为过程，并深入代码内部的结构，解析有问题的会话。

- 测试管理工具：测试管理工具用于对测试进行管理。一般而言，测试管理工具对测试需求、测试计划、测试用例、测试实施进行管理，并且测试管理工具还包括对缺陷的跟踪管理。测试管理工具能让测试人员、开发人员或其他的 IT 人员通过一个中央数据仓库，在不同的地方就能交互信息。测试管理工具将测试过程流水化，从测试需求管理到测试计划、测试日程安排、测试执行到出错后的错误跟踪，实现了全过程的自动化管理。测试管理工具的代表有 TestDirector、TestManger、TrackRecord 等。
- 测试辅助工具：这些工具本身并不执行测试，例如它们可以生成测试数据，为测试提供数据准备等。

2．自动化测试应用策略

随着软件测试地位的逐步提高，测试的重要性逐步显现，测试工具的应用已经成为了普遍的趋势。总的来说，测试工具的应用可以提高测试的质量、测试的效率。但是在选择和使用测试工具的时候，我们也应该看到，在测试过程中，并不是所有的测试工具都适合我们使用，同时，有了测试工具并且会使用测试工具并不等于测试工具真正能在测试中发挥作用，因此，如何确定自动化测试策略显得至关重要。应用测试工具的目的很明确，一般而言，在测试过程中应用测试工具主要有以下几个目的。

- 提高测试质量；
- 减少测试过程中的重复劳动；
- 实现测试自动化，解决手工测试不能解决的问题。

为了更好地达到测试目的，在信息系统中应用自动化测试需要考虑以下问题。

- 选择合适的自动化测试工具：面对众多不同用途的测试工具，如何正确地选择合适的测试工具，是能否正常实施自动化测试的前提，我们在选用工具的时候，建议从以下几个方面来权衡。

① 功能：功能当然是我们最关注的内容，选择一个测试工具首先就是看它提供的功能。当然，这并不是说测试工具提供的功能越多越好，在实际的选择过程中，适用才是根本。“钱要花在刀刃上”，为不需要的功能花费金钱是不明智的行为。事实上，目前市面上同类的软件测试工具之间的基本功能都是大同小异的，各种软件提供的功能也大致相同，只不过有不同的侧重点。例如，同为白盒测试工具的 Logiscope 和 PRQA 软件，它们提供的基本功能大致相同，只是在编码规则、编码规则的定制、采用的代码质量标

准方面有不同。除了基本的功能之外，以下的功能需求也可以作为选择测试工具的参考。

- 报表功能：测试工具生成的结果最终要由测试人员进行解释，而且，查看最终报告的人员不一定对测试很熟悉，因此，测试工具能否生成结果报表，能够以什么形势提供报表是需要考虑的因素（标准符号有些混乱）。
- 测试工具的集成能力：引入测试工具是一个长期的过程，应该是伴随着测试过程改进而进行的一个持续的过程。因此，测试工具的集成能力也是必须考虑的因素，这里的集成包括两个方面的意思，首先，测试工具能否和开发工具进行良好的集成；其次，测试工具能够和其他测试工具进行良好的集成。
- 操作系统和开发工具的兼容性。测试工具可否跨平台，是否适用于公司目前使用的开发工具，这些问题也是在选择一个测试工具时必须考虑的问题。

② 价格：除了功能之外，价格就应该是最重要的因素了，作为一个测试工程师，在选择购买测试工具时，应该具有成本意识，必须利用有限的资金满足企业对测试工具的大多数需求。

③ 测试工具的长期投资考虑：测试工具引入的目的是测试自动化，引入工具需要考虑工具的连续性和一致性，也就是说，对测试工具的选择必须有一个全盘的考虑，分阶段、逐步的引入测试工具。

- 确定测试工具的应用时机：购买了测试工具以后，如何让测试工具真正发挥作用，是应用自动化测试的关键。任何测试工具都有其应用范围，也许我们具备不同的测试工具，那么在不同的软件工程阶段，我们应该有计划地去使用相应的测试工具，并将测试工具的使用明确定义进公司的开发流程。例如，在单元测试阶段，我们应该重点采用白盒测试工具，当软件产品的功能以及用户界面基本确定和逐步实现后，则可以考虑开始使用功能测试工具。集成测试阶段，则可以引入负载压力测试工具，对系统可能承受的负载压力进行测试与评估，并辅以相应的资源，使用监控工具进行故障定位等。
- 确定测试重点：对于一些测试项目，尤其是在测试时间有限的情况下，比如，执行一次性能测试，我们必须能够确定被测项目的主要应用和关键步骤，应该对那些质量要求较高并且风险大的部分进行重点测试，例如在金融领域，对于那些每天管理数百万、数千万人民币流动的系统，需要特别对其硬件、软件的安全可靠性、可用性进行测试。
- 确定测试目标和指标：针对不同的软件，其软件质量要求的等级和目标是不一样的，通过测试工具可以更好地验证系统设计是否达到了预期目标，因此，在正式开始测试前，我们应该能够清楚地了解测试预期目标。
- 充分利用测试工具的优势：每个测试工具都有自己独特的实现技术，对于同一



个测试项目，测试工具可能也提供了多种测试方案供选择，比如脚本录制过程中协议的选择，回放过程中用户并发模拟机制和方式的选择等，只有充分利用了测试工具提供的这些技术，才可能更好、更真实地测试应用系统的实际质量。

- 加强对测试工程师的技能培训，测试工具的使用者必须对测试工具非常了解。在这方面，有效的培训是必不可少的。测试工具的培训是一个长期的过程，不是通过一两次讲课的形式就能达到良好的效果的。而且，在实际使用测试工具的过程中，测试工具的使用者可能还存在着这样那样的问题，这也需要有专家负责解决，否则的话，对于测试工具使用者的积极性将造成很大的打击。

2.11.4 功能自动化测试

1．概述

传统的手工测试是测试人员执行测试用例，然后将测试结果和预期结果相比较并记录测试结果。然而，随着软件工程的规模越来越大，软件产品的功能越来越复杂，同时，软件的更新换代也越来越频繁，软件测试部门的工作难度越来越大，手工测试已经跟不上这种发展趋势了。

功能自动化测试工具可以帮助测试工程师自动处理测试开发到测试执行的整个过程中的问题。你可以创建可修改且可复用的测试脚本，甚至可以在下班后让计算机自动执行脚本，从而减少劳动量，提高测试效率。

功能测试自动化工具的主要功能，就是为了确保应用能够按照预期设计执行而将业务处理过程记录到测试脚本中。当应用被开发完成或应用升级时，测试工具支持测试脚本的编辑、扩展、执行和报告测试结果，并且保证测试脚本的可重复使用，贯穿于应用的整个生命周期。

当一个应用开发完毕后，程序界面基本定型，这个时候，针对该应用的自动测试应该展开。自动测试的引入，大大提高了测试的效率和测试的准确性，而且测试人员一次设计的脚本，可以在软件生命周期的各个阶段重复使用，尤其在软件交付后，随着企业的发展，你的应用就会随之在数量和范围上增长。为了满足业务的需求，应用的改变会很频繁，对于这些需求，将可以通过小范围修改测试工具录制的脚本来完成。对于功能测试工具的使用，比较重要的是测试规划问题。如何规划一次录制，使它具有良好的可扩展性、重用性，整个脚本能够有清晰的层次和最大限度地适应以后程序的修改，这些也是在实际工程中用户最普遍遇到的问题，它的实施就需要有经验的软件测试人员介入并结合应用来进行具体分析。

2．测试原理

功能自动化测试工具基本上都是采取录制回放的方式来模拟用户的实际操作。当你

在软件操作中点击图形用户界面上的对象时，测试工具会用一种类 C 或者其他的脚本语言（TSL）生成一个测试脚本，该脚本记录了你的操作过程，然后测试工具就可以回放刚才的操作过程。当然你也可以手工编程生成这个脚本。通常情况下，测试工具采取两种录制模式。

（1）环境判断模式

这种模式根据你选取的图形用户界面对象（如窗体、清单、按钮等），把你对软件的操作动作录制下来，并忽略这些对象在屏幕上的物理位置。每一次你对被测软件进行操作，测试脚本语言会记录并描述你选取的对象和你的操作动作。当你进行录制时，测试工具会对你选取的每个对象做惟一描述并写入相应的文件中。当软件用户界面发生变化时，你只需更新特定的对象记录文件。这样一来，环境判断模式的测试脚本将非常容易地被重复使用。执行测试只需要回放测试脚本。回放时，测试工具从指定文件中读取对象描述，并在被测软件中查找符合这些描述的对象并模拟用户使用鼠标选取该对象、用键盘输入数据的操作。

（2）模拟模式

这种模式记录鼠标点击、键盘输入和鼠标在二维平面上（x 轴和 y 轴）的精确运动轨迹。执行测试时，测试工具让鼠标根据轨迹运动。这种模式对于那些需要追踪鼠标运动的测试非常有用，例如画图软件等。

不论测试工具采用的是哪种录制模式，通常情况下，其实施测试必须经历的几个操作步骤如下。

- 创建脚本：你可以通过录制、编程或两者同用的方式创建测试脚本。测试工具可以自动记录你的操作并生成所需的脚本代码，你还可以直接修改测试脚本以满足各种复杂测试的需求。录制测试时，在需要检查软件反应的地方插入检查点（checkpoint）。你可以插入检查点来检查 GUI 对象、位图（bitmap）和数据库。在这个过程中，测试工具会自动捕捉数据，并将该数据作为期望结果储存下来。例如，在创建测试时，可以设定哪些数据库表和记录需要检测，在测试运行时，测试程序就会自动核对数据库内的实际数值和预期的数值是否一致。
- 调试脚本：脚本录制或编辑结束后，你可以先在调试模式下运行脚本。并可以设置中断点（breakpoint）来监测变量，控制对象识别和隔离错误。
- 执行测试：脚本调试结束后，便可以在检验模式下测试被测软件。运行测试时，测试工具会自动操作应用程序，就像一个真实的用户根据业务流程执行着每一步的操作。此时，测试工具在运行脚本过程中如果遇到了检查点，就把当前数据和事先捕捉并保存的期望值进行比较。如果发现有不符合，就记录下来作为



测试结果。在具体的测试过程中，为了全面地测试一个应用程序，需要使用不同类型的数据来测试。一般情况下，测试工具都能提供动态数据处理及参数化技术，可以用参数去代替定值，从而真实地反映多个用户行为。以一个订单输入的流程为例，你可能希望把订单号或客户名称作为可变栏，用多套数据进行测试。使用数据驱动向导，你可以选择订单号或客户名称用数据表格文件中的哪个栏目的数据替换。你可以把订单号或客户名称输入数据表格文件，或从其他表格和数据库中导入。数据驱动测试不仅节省了时间和资源，又提高了应用的测试覆盖率。

- 结果分析：每次测试结束，测试工具都会把测试情况显示在测试结果报告中。测试结果报告会详细描述测试执行过程中发生的所有主要事件，如检查点、错误信息、系统信息或用户信息。如果在检查点有不符合的情况被发现，你可以在测试结果窗口查看预期结果和实际测试结果。如果是位图不符合，你也可以查看用于显示预期值和实测结果之间差异的位图。如果由于测试中发现错误而造成测试运行失败，你可以直接从测试结果中查看有关错误信息。

2.11.5 负载压力自动化测试

1. 概述

当一个企业自己组织力量或委托其他的软件公司开发了一套应用系统的时候，尤其是以后在生产环境中实际使用起来时，用户往往会产生疑问，这套系统能不能承受大量的并发用户同时访问，随着生产数据的不断累积，系统的响应处理能力是不是会明显下降直至用户不能接受。这类问题最常见于采用联机事务处理（OLTP）方式的数据库应用、Web 应用和视频点播应用等系统。

比如，电信计费软件，众所周知，每月 20 日左右是市话交费的高峰期，全市几千个收费网点同时启动，收费过程一般分为两步，首先要根据用户提出的电话号码来查询出其当月产生的费用，然后收取现金并将此用户修改为已交费状态。一个用户看起来简单的两个步骤，当成百上千的终端同时执行这样的操作时情况就大不一样了，如此众多的交易同时发生，对应用程序本身、操作系统、中心数据库服务器、中间件服务器、网络设备的承受力都是一个严峻的考验。决策者不可能在发生问题后才考虑系统承受力。预见软件系统的并发承受力，这是在软件测试阶段就应该解决的。

如何模拟实际情况呢？找若干台电脑和同样数目的操作人员在同一时刻进行操作，然后拿秒表记录下反应时间？这样的手工作坊式的测试方法不切实际且无法捕捉程序内部的变化情况。这就需要负载压力测试工具的辅助。

那么，什么是负载测试和压力测试呢？负载测试是为了证明在与产品（预期）规模

等同的数据库中处理给定的事务请求的容量下，系统功能与性能是否与需求规格说明中规定的，可接受的响应时间一致的测试过程。而压力测试则是使客户机在大容量情况下运行的测试过程，目的是查看应用将在何时何处出现中断，即识别系统的薄弱环节。压力测试中可能暴露的系统缺陷有内存泄漏、系统资源过量消耗、磁盘空间用完等。

负载压力测试工具可以记录下客户端的操作，并以脚本的方式保存，然后建立多个虚拟用户，在一台或几台 PC 机上模拟上百或上千虚拟用户同时操作的情景，同时记录下每一事务处理的响应时间、中间件服务器资源使用、数据库负载等，测试工程师可以根据测试结果分析系统瓶颈。

在各种类型的并发测试中，基于 Web 的应用占了很大的比例，现在有相当数量的联机事务处理（OLTP）类型系统采用 Web 方式，还有一些网站，对并发连接的数量和自己网站对大量用户访问的支持能力，都表示出了相当程度的关心。对于负载压力测试工具，它提供了对 Web 页面压力测试的完整解决方案，包括用户模拟、Web 服务器监控、页面每秒钟点击率统计、单独页面加载时间分析等各种专门针对 Web 的特性。

随着服务器端处理任务的日益复杂以及网站访问量的迅速增长，服务器性能的优化也成了非常迫切的问题。在优化之前，最好能够测试一下不同负载条件下服务器的性能表现。定位性能瓶颈所在，是设计性能改善方案之前的一个至关紧要的步骤。

负载测试是任何 Web 应用的开发周期中一个重要的步骤。如果你在构造一个为大量用户服务的应用，搞清楚你的产品配置能够承受多大的负载非常重要。如果你在构造一个小型的 Intranet 网站，测试能够暴露出最终会导致服务器崩溃的内存漏洞、资源竞争等情况。但是在实际的开发过程中，要按照实际投入运行的情况，组织成千上万的用户来进行压力测试，无论从哪个方面看，都是不现实的。而且一旦发现了问题，不仅需要重复地进行这种耗费巨大的测试，而且问题不容易重现，不能方便地找出性能的瓶颈所在。而使用测试工具进行压力测试就不会存在这种情况。

无论是哪种情形，花些时间对应用进行负载压力测试可以获得重要的基准性能数据，为未来的代码优化、硬件配置以及系统软硬件升级带来方便。即使经费有限的开发组织也可以对它们的网站进行负载压力测试，因为有些测试工具是可以免费下载的。

在测试阶段使用负载压力测试工具进行测试，还可以模拟数据库死锁情况，结合压力分析 SQL 效率，优化应用程序和数据库配置等工作，使软件系统更加健壮和高效。这样的实例也很多，比如有公司在测试某省的大型电信业务网上受理系统时，200 个并发用户同时联机时速度正常，但当达到用户量达到 500 个的时候，受理速度明显变慢，通过监控发现 Web 服务器的流量有所降低，而表空间对应的数据文件中发生的磁盘物理读的次数却大于正常水平，最后通过诊断确定有部分复杂的 SQL 查询（如大表连接操作，



嵌套查询等）没有利用合适的索引和采用最优的解释方案，而造成全表扫描。而且数据库配置参数 DB_BLOCK_BUFFERS 太小，不能适应 500 个用户或更大规模的并发情况。经过测试人员和开发人员对系统的共同调整，再次测试的时候一切恢复正常，500 个用户的并发测试顺利通过。

2．测试原理

负载压力测试工具基本上都是采取录制回放的方式来模拟用户的实际操作的，而且测试工具一般都会有一个后台代理进程，通过该代理进程，测试工具可以监视并获取在各种通信协议下应用系统的客户与服务器端的通信信息，测试工具会用一种类 C 或者其他的脚本语言（TSL）生成一个测试脚本，该脚本记录了你对服务器的请求过程，然后测试工具就可以回放刚才的访问过程，接收服务器的响应，当然你也可以用手工编程生成这个脚本。

通常情况下，实施负载压力测试的工作示意图如图 2-21 所示。

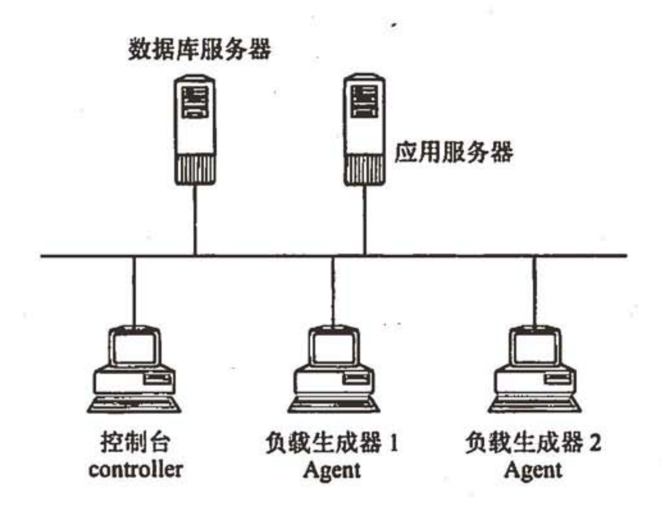


图 2-21　实施负载压力测试的工作示意图

当被测系统运行时，脚本生成器会自动获取系统客户端与服务器的通信信息并转换成测试工具可以识别的脚本，测试工具通过控制台将测试脚本分发到其他负载生成器上以模拟多用户对服务器的并发访问，同时，控制台还可以通过服务器上开启的远程 RPC 服务，获取相关的资源使用信息，最后可以收集测试数据。

在进行测试脚本录制与分配的过程中，应遵循如下几个原则。

- 脚本越小越好。就像编写程序代码一样，不要太长，尽量做到一个功能一个脚本。如果那些功能是连续有序的，必须先做上一个，才能执行下一个，那就只

好放在一起了。

- 选择负载压力最高的业务功能进行测试。有些人总希望能够测试几乎所有的功能，但事实上，这种想法不可行，因为测试是需要投入时间和金钱的，我们应该结合用户实际的使用情况，选择负载压力最高的业务功能进行测试，以满足性能测试的需求，达到测试的预期目的。
- 选择所需要的操作进行录制。例如，需要测试服务器承受负载压力的性能，某些客户端的操作不会对后台服务器产生负载压力，就可以不录制。比如一些查询操作，选择查询条件的页面可以不录制，但对于一些页面有可能要与后台服务器传递参数，就需要录制了。如何确定哪些操作需要录制，一是可以找开发人员了解清楚程序设计结构和运行模式，二是可以靠自己的经验，熟能生巧当然不会错。

由于负载压力测试工具是一种预测系统行为和性能的负载测试工具。它需要模拟成百上千甚至上万的用户同时操作应用系统，实施并发负载及实时性能监测。通常情况下，测试工具模拟多用户并发访问可以有以下两种方式。

① 进程回放模式：模拟多进程运行方式，即客户端与服务器的访问采用进程方式，每一个虚拟用户通过一个进程建立与服务器的通信连接并访问。

② 线程回放模式：模拟多线程运行方式，即客户端与服务器的访问采用线程方式，每一个虚拟用户通过一个线程建立与服务器的通信连接并访问。

不论测试工具采用的是哪种录制回放模式，其必须经历的几个操作步骤如下。

- 协议选择：如前所述，测试工具都是通过获取在各种通信协议下应用系统的客户端与服务器端的通信信息并进一步来实现负载压力测试的，所以首先要确定被测应用系统客户端与服务器端的通信所使用的协议类型。一般而言，B/S 系统选择 Web（Http/Html），两层 C/S 系统则根据 C/S 结构所用到的后台数据库来选择不同的协议，协议的选择请参考有关章节。
- 创建测试脚本：选择好相应的录制协议以后，就可以启动脚本录制工具，通过测试工具的代理进程获取应用系统客户端和服务器端的通信信息，测试工具可以自动记录客户端对服务器端的访问操作并自动转换成所需的脚本代码，还可以直接编辑测试脚本以满足各种复杂测试的需求。
- 参数化测试数据。对于创建的测试脚本，你可以利用数据池技术对其中的某些数据参数化，从而更好地模拟真实应用访问。以一个订单输入过程为例，参数化操作可将记录中的常量数据，如订单号和客户名称，由变值来代替，以更好地模拟多个实际用户的操作行为。
- 创建虚拟用户，设定负载方案：由于负载压力测试的目的就是要模拟多用户并



发访问应用系统，所以在开始测试之前就应该设计好需要模拟的虚拟用户数，然后使用测试工具控制台设定负载方案。

- 执行测试：设定了相应的负载测试方案后，就可以开始测试了，测试过程中，测试工具会自动记录测试结果，包括事务处理的响应时间，服务器的资源占用情况等。
- 结果分析：测试结束后，测试工具会收集汇总所有的测试数据并生成测试结果报告，这些数据主要包括交易性能数据（如响应时间等）、服务器资源占用情况、网路设备和数据库的实时性能数据等。通过这些数据就可以从客户端、服务器端以及网络三方面来评估系统组件的运行性能，从而定位应用系统存在的主要问题，为系统改进做准备。

第 3 章　软件质量与评价（软件测试标准）

对软件质量进行评估是软件测试的一个重要目的。软件测试人员必须深刻理解软件质量的定义和度量原理。

3.1　质量的定义

想提高质量，就必须给出质量定义并测量它。大众化的观点和含糊其词的定义，不利于质量工程的开展。质量是多维的概念，包括：实体、实体的属性和对实体的观点。

早在 1970 年，Juran 和 Gryna 把质量定义为“适于使用”。Crosby（1979 年）将质量定义为“符合需求”。“符合需求”隐含着需求必须明确地说明的意思；而“适于使用”隐含了顾客的需求和预期。顾客的需求和预期包括了产品和服务是否适合顾客使用。

GB/T 6583 - ISO 8404（1994 版）《质量管理与质量保证术语》对质量的定义是“反映实体满足明确的和隐含的需要的能力的特性的总和”。作为软件质量，在 GB/T 18905-ISO 14598（1999 版）《软件工程 产品评价》中也是这样定义的：“实体特性的总和，满足明确或隐含要求的能力”。

3.2　测度与度量

一个实体的质量好坏是需要测量的，而测量就需要首先建立质量指标体系或质量模型，然后使用特定测量方法才能实施测量。测度的运用是建立测量方法的依据，也是解决软件质量测量的关键。

测度是把数字和符号分配给现实世界实体的属性，根据明确定义规则来定义它们 Fenton[FEN91]。运用测度，人们能较好地理解建立质量模型的属性，并评价所建立的工程化产品或系统质量。虽然软件技术度量不是绝对的，但测度的建立提供了基于一套清晰定义规则的系统方法来评价软件质量。软件质量模型和评价技术就是基于运用测度的建立原则发展起来的。

软件工程中使用的“度量”有多种含义，在软件质量中用于测量的一种量化的标度和方法即为“测度”，而名词的“度量”用来指测量的结果。



3.3 软件质量模型

从测量的角度看，影响软件质量的因素可以分为两大类：可直接测量（如每个功能点的错误）和间接度量（如可用性、可维护性）。每种类型测度都必须发生。

早期的软件质量模型是 1977 年 McCall 和他的同事建立的，提出了影响质量因素的有用的分类。McCall 质量模型如图 3-1 所示，集中在软件产品的三个重要方面：操作特性（产品运行）、承受可改变能力（产品修订）、新环境适应能力（产品变迁）。

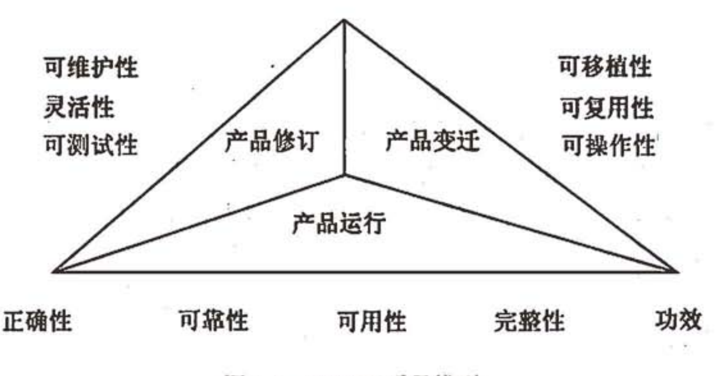


图 3-1 McCall 质量模型

1978 年 Boehm 和他的同事们提出了分层结构的软件质量模型，除包含了用户的期望和需要的概念，这一点与 McCall 相同之外，还包括了 McCall 模型中没有的硬件特性。Boehm 质量模型如图 3-2 所示。

Boehm 模型始于软件的整体效用，从系统交付后涉及不同类型的用户考虑。第一种用户是初始顾客，系统做了顾客所期望的事，顾客对系统非常满意；第二种用户是要将软件移植到其他软硬件系统下使用的客户；第三种用户是维护系统的程序员。三种用户都希望系统是可靠有效的。因此，Boehm 模型反映了对软件质量的理解，即软件做了用户要它做的；有效地使用系统资源；易于用户学习和使用；易于测试与维护。

20 世纪 90 年代早期，软件工程组织试图将诸多的软件质量模型统一到一个模型中，并把这个模型作为度量软件质量的一个国际标准。国际标准化组织 1991 年颁布了 ISO 9126-1991 标准《软件产品评价—质量特性及其使用指南》。我国也于 1996 年发布了同样的软件产品质量评价标准 GB/T 16260-1996。它是一个分层质量模型，有 6 个影响质量的特性。如图 3-3 所示说明了质量特性与质量子特性的层次结构。

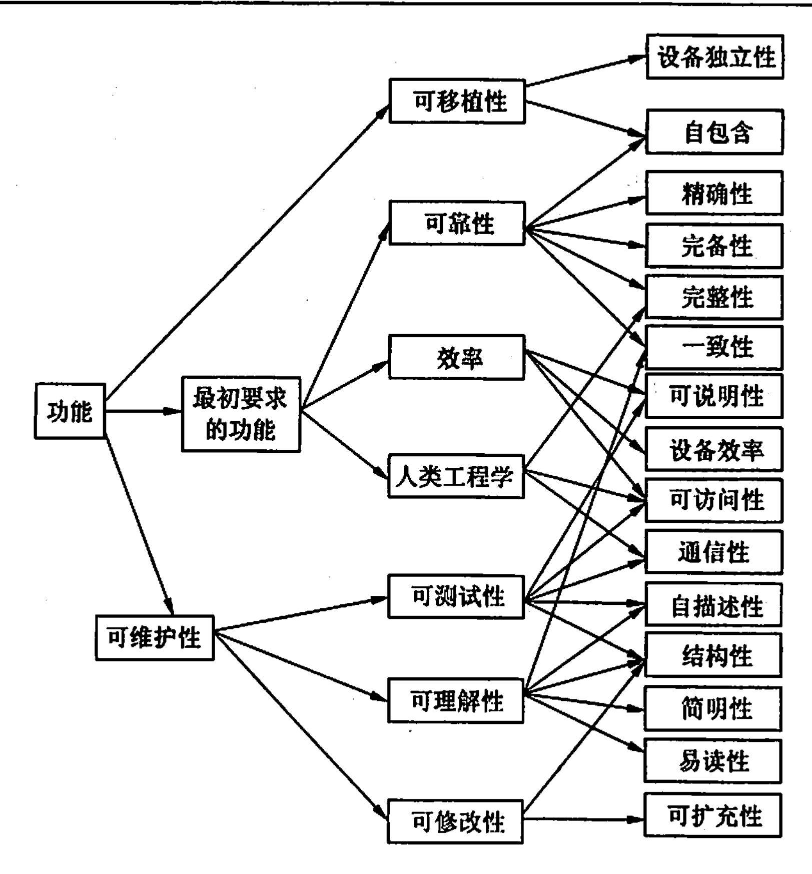


图 3-2　Boehm 质量模型

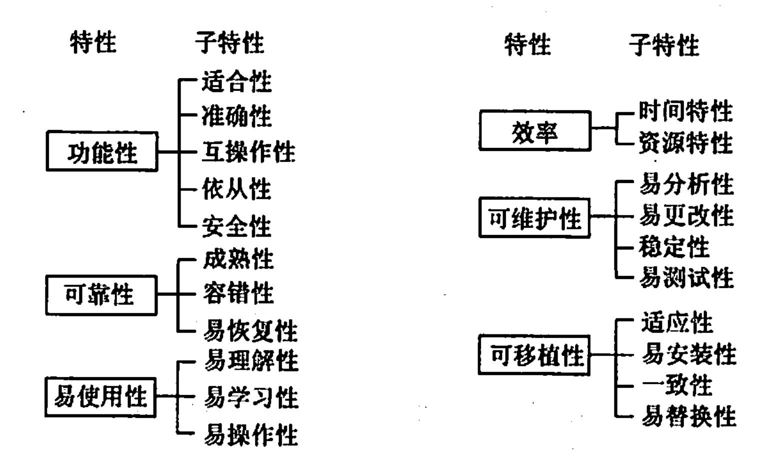


图 3-3　ISO 9126 质量模型



标准的软件质量测度是这样建立的：软件质量模型分三个层次，第一层有6个影响软件质量的主要因素，在标准中称之为“质量特性”。而每个质量特性又可以通过第二层的若干个子特性测量，第二层的每个子特性在评价时要定义并实施若干个度量。当时，ISO 9126 资料性的附录中给出了21个子特性。

ISO 9126 的出发点在于使软件最大限度地满足用户的明确的和潜在的需求。这6个质量特性最大可能地涵盖了其他早期质量模型中所有的因素，而且彼此交叉最小。软件质量特性与子特性的定义是从用户的角度、开发者的角度和管理者的角度全方位出发考虑的。因此，ISO 9126-1991 在当时是最为先进、严格的质量模型，它划时代地统一了十几年来国际上推出的各种质量模型。

3.4 标准的发展

随着信息技术应用的不断增长，关键的计算机系统的数量也在增长。这些系统包括：安全、生活、经济以及保密方面的关键系统。这些系统的软件质量尤其重要，因为软件的故障可能导致非常严重的后果。

纵观整个软件工程的历史，提高软件质量已成为最重要的目标。评价软件产品的质量对获取和开发满足质量需求的软件是不可缺少的。各种软件质量特性的相关重要性取决于作为整体一部分的系统的任务和目标，需要评价软件产品以判断其相关的质量特性是否满足系统的需求。

到 1999 年，国际软件工程标准化组织将软件“产品评价”与“产品质量”分成两个标准，“产品评价”注重软件质量评价的支持和评价过程；“产品质量”注重软件本身的质量度量模型。

软件质量评价的基本部分包括：质量模型、评价方法、软件的测量和支持工具。要想开发好的软件，就要规定质量需求，策划、实现和控制软件质量保证过程，应评价中间产品和最终产品。要达到评价软件质量的目的，应用有效的度量方法测量软件的质量属性是非常必要的。

GB/T 18905—2002（ISO 14598—1999）《软件工程 产品评价》系列标准为软件产品质量的测量、评估和评价提供了方法。它所描述的既不是软件生产过程的评价方法，也不是预算成本的方法（软件产品的质量测量当然可以用于这两个目的）。

因为质量特性和相关的度量不仅可用于软件产品评价，而且也可用于质量需求的定义以及其他用途，国际软件工程标准化组织颁布了 ISO 14598—1999《软件工程 产品评价》，而在 2001 年修订了 ISO 9126—1991 标准。

所以，早期的 GB/T 16260—1996（ISO 9126—1991）《软件产品评价—质量特性及

其使用指南》已经被两个相关的由多部分组成的标准：GB/T 18905—2002（ISO 14598—1999）《软件工程 产品评价》和 GB/T 16260—2003（ISO 9126—2001）《软件工程 产品质量》所取代。

3.5 GB/T 18905 产品评价

3.5.1 GB/T 18905 基本组成

GB/T 18905—2002《软件工程　产品评价》。该系列标准由以下 6 个部分组成。

- GB/T 18905.1《软件工程　产品评价》第 1 部分，概述。
- GB/T 18905.2《软件工程　产品评价》第 2 部分，策划和管理。
- GB/T 18905.3《软件工程　产品评价》第 3 部分，开发者用的过程。
- GB/T 18905.4《软件工程　产品评价》第 4 部分，需方用的过程。
- GB/T 18905.5《软件工程　产品评价》第 5 部分，评价者用的过程。
- GB/T 18905.6《软件工程　产品评价》第 6 部分，评价模块的文档编制。

GB/T 18905.1 概述了软件产品评价的过程，提供了评价需求和指南；GB/T 18905.2 和 GB/T 18905.6 是关于公司或部门级的评价管理和支持；GB/T 18905.3、GB/T 18905.4 和 GB/T 18905.5 给出了项目级的评价需求和指南。如图 3-4 所示给出了这些标准和技术报告之间的关系。

3.5.2 评价者用的过程（GB/T 18905.5）

1．开发者用的过程

计划开发新产品或增强现有的产品，以及打算利用他们自己的技术人员进行产品评价的组织应使用 GB/T 18905.3。这部分主要强调使用那些能预测最终产品质量的指标，这些指标将通过度量在生存期期间开发的中间产品来得到。

2．需方用的过程

计划获取或复用某个已有的软件产品或预先开发的软件产品的组织应使用 GB/T 18905.4。该部分可用来决定接受产品或者从众多可选产品中选择某个产品（产品可以是自包含的，或是系统的一部分，或者是较大产品的一部分）。

3．评价者用的过程

对软件产品执行独立评估的评价者应使用 GB/T 18905.5。这种评价是应开发者、需方或其他方的请求来进行的。这部分将由那些执行独立评价的人员使用，他们通常为第三方组织工作。



3.5.3 关于评价支持

上述每个评价过程的标准都能与 GB/T 18905.2（策划和管理）和 GB/T 18905.6（评价模块的文档编制）结合起来使用（如图 3-4 所示）。

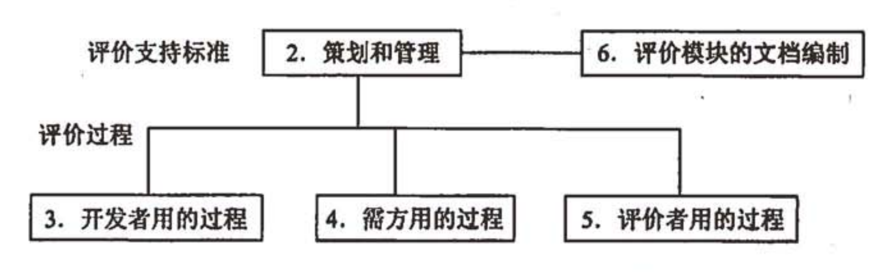


图 3-4 评价过程与评价支持标准的关系

1. 策划和管理

GB/T 18905.2 为策划和管理部分,包含对软件产品评价的支持功能的需求和指南。这种支持与策划和管理软件评价过程及相关的活动有关,包括组织内评价专业知识的开发、获取、标准化、控制、转换和反馈。可供管理者制定一个定量的评价计划。

2. 评价模块

GB/T 18905.6 为编制评价模块的文档提供指南。这些模块包括质量模型的规范（即特性、子特性和相应的内部或外部度量），与模型计划的应用有关的数据、信息和与模型的实际应用有关的信息。每种评价都应选择适当的评价模块。在某些情况，还有必要开发新的评价模块，以供组织用来产生新的评价模块。

3.5.4 通用评价过程

软件产品的一般评价过程是：确立评价需求，然后，规定、设计和执行评价，如图 3-5 所示。

3.5.5 评价需求

软件质量评价的目的是为了直接支持开发和获得能满足用户和消费者要求的软件。最终目标是保证产品能提供所要求的质量，即满足用户（包括操作者、软件结果的接受者，或软件的维护者）明确和隐含的要求。

（1）评价中间产品质量的目的

- 决定（是否）接受分包商交付的中间产品；
- 决定某个过程的完成，以及何时把产品送交下个过程；

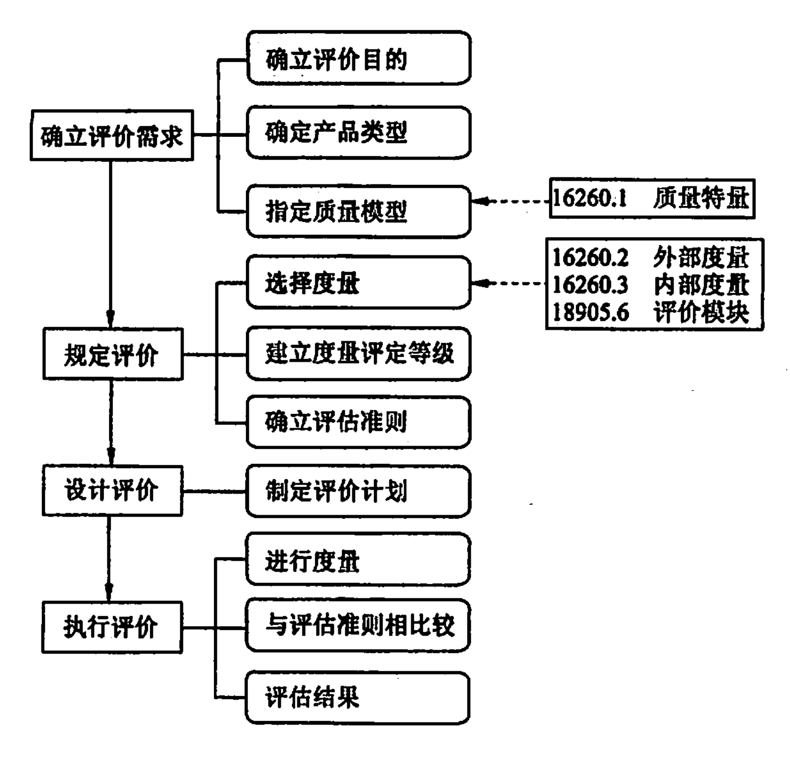


图 3-5　软件评价过程

- 预计或估计最终产品的质量；
- 收集中间产品的信息以便控制和管理过程。

（2）评价最终产品质量的目的

- 决定（是否）接受产品；
- 决定何时发布产品；
- 与相互竞争的产品进行比较；
- 从众多可选的产品中选择一种产品；
- 使用产品时评估产品积极和消极的影响；
- 决定何时增强或替换该产品。

3.5.6　确定要评价产品的类型

要评价的中间或最终软件产品的类型取决于所处的生存周期的阶段和评价的目的，如图 3-6 所示。

需求

真实世界

要求

使用质量

操作

外部度量

使用并反馈

确定

指示

规格说明

系统集成和测试

系统行为

外部质量要求

外部质量

外部度量

确定

确认

指示

设计和开发

软件属性

内部质量要求

内部质量

内部度量

验证

图 3-6 软件生存周期中的质量

3.5.7 度量之间的关系

将软件的内部质量属性与外部质量需求联系起来是十分重要的，使得开发中的软件产品（中间和最终的软件产品）的质量特性可以根据最终系统使用质量需求来进行评估。内部度量的值很低，除非有证据表明它与外部质量有关。如图 3-7 所示。

3.5.8 规定质量模型

软件评价所用的质量模型通常代表软件质量属性的总体，这些质量属性用特性和子特性的分层树结构进行分类。该结构的最高级由质量特性构成，最低级由软件质量属性构成。GB/T 16260.1 提供了一个通用模型，它定义了 6 种软件质量特性，包括功能性、可靠性、易用性、效率、可维护性和可移植性。这些特性还能进一步被分解为具有可测量属性的子特性。在特定的使用环境下，质量特性的组合效应被定义为使用质量。

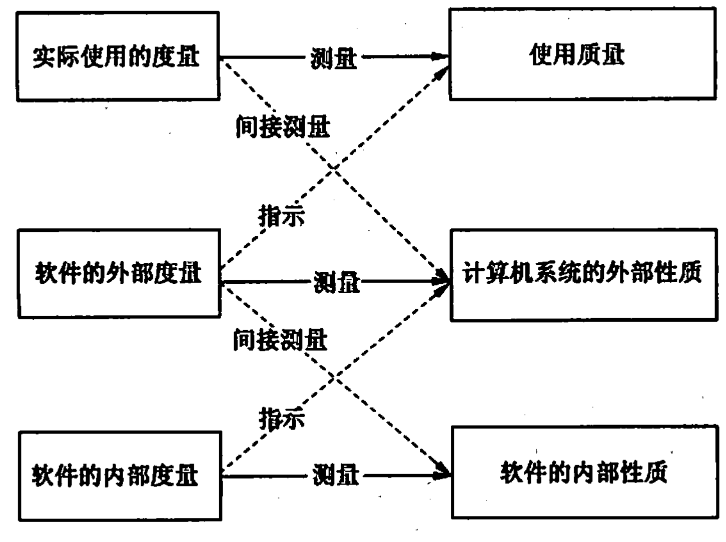


图 3-7　度量之间的关系

软件产品的内部质量属性是软件产品的可测量的性质，这些性质影响产品满足明确和隐含要求的能力。可以用一个或更多的属性来评估一个特定的软件质量特性或子特性，如图 3-8 所示。

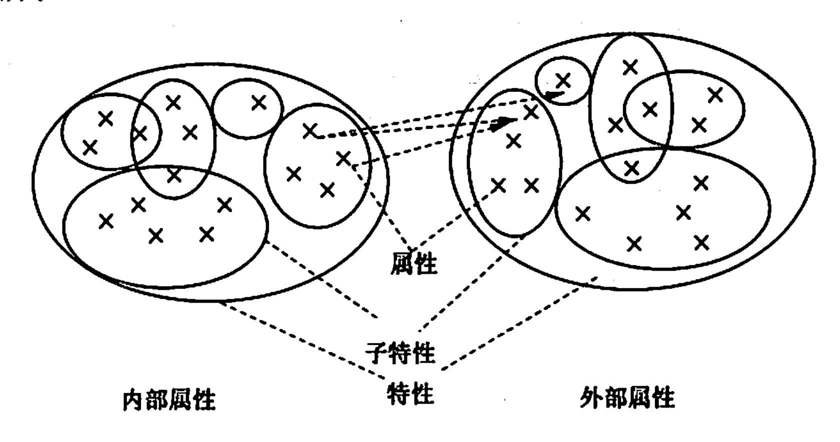


图 3-8　质量特性、子特性和属性

3.5.9　规定评价

（1）选择度量

重要的是软件产品的测量要能既简单又经济地进行，而且测量结果要易于使用。对



许多软件的测量可以方便地用某种工具来进行，也可以打包成一个评价模块（见 GB/T 18905.6）。

质量特性的定义方式不允许对它们进行直接测量。需要建立与软件产品特性相关的度量。与某个质量特性相关的每个可量化的软件内部属性和每个可量化的软件外部属性，与其软件环境进行相互作用，能被确立为一种度量。

度量可以随环境和应用度量的开发过程阶段的不同而有所区别。用于开发过程的度量应与用户观点的度量有关，因为从用户视角出发的度量是至关重要的。

（2）测量的种类

评价有两个主要目的。

- 确定问题以便解决问题；
- 与可替换的产品进行比较，或对照需求比较产品质量。

所需的测量种类取决于评价的目的。如果主要目的是为了了解和纠正缺陷，可以对软件采取多种测量，以便监视和控制改进。

（3）确立度量评定等级

可量化的特征可以用度量质量的方法进行定量的测量。其结果是，将测量值映射到某一标度上。这个值本身并不表示满意的等级，因此，这一标度必须根据需求的不同满意度级别分成不同的范围。例如，将标度分成两类：满意和不满意；将标度分成四类：针对已有产品或可替换产品的当前级、最差级和计划级划分成四类，即超出要求、目标范围、可接受的最低限度、不可接受。定义当前级是为控制新系统不因当前状况而恶化；计划级是指一旦资源可利用，产品即可获得；最差级是指万一产品不符合计划级时用户的可接受边界如图 3-9 所示。

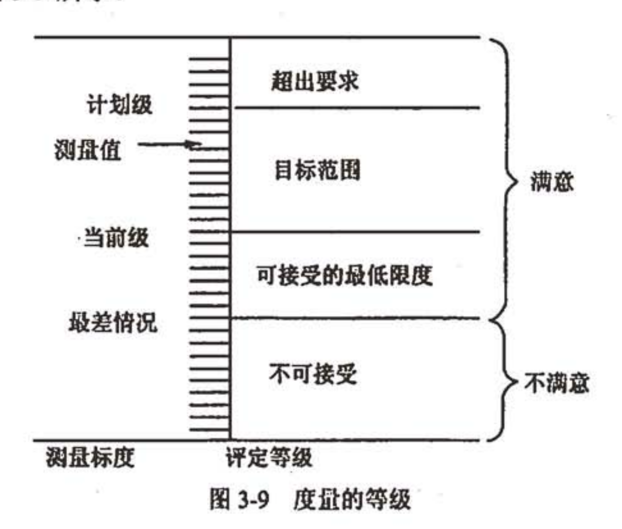


图 3-9 度量的等级

（4）确立评估准则

软件质量需求规格说明应使用定义良好的、适当的质量模型来表示。为此，除非有特殊原因需使用其他模型外，应使用 GB/T 16260.1 中的质量模型和定义。

为了评估产品质量，需要总结针对不同特性的评价结果。评价者应为此准备一个规程，其中，对不同的质量特性使用不同的评价准则，每个质量特性又以数个子特性或子特性的加权组合来说明。规程通常还包括时间和成本等有助于在特定环境下评估软件产品质量的其他方面。

3.6　GB/T 16260.1 产品质量

3.6.1　基本组成

GB/T 16260—2003《软件工程　产品质量》。该系列标准由以下 4 部分组成。

- GB/T 16260.1《软件工程　产品质量》第 1 部分，质量模型。
- GB/T 16260.2《软件工程　产品质量》第 2 部分，外部度量。
- GB/T 16260.3《软件工程　产品质量》第 3 部分，内部度量。
- GB/T 16260.4《软件工程　产品质量》第 4 部分，使用质量度量。

3.6.2　标准概述

1．标准的变化

国际上在 2001 年对软件质量特性评价标准 ISO 9126 进行了修订，保留了原来的 6 个软件质量特性,定义了一个通用的质量模型，并给出了度量的例子。与原标准相比，其主要区别在于：

- 质量特性中增加了使用质量特性；
- 质量度量分为外部度量、内部度量和使用质量度量；
- 子特性作为标准的一部分，在原版标准资料性附录中的子特性基础上增加了一些；
- 删除了评价过程内容（已在 ISO 14598 中进行了说明）；
- 与 ISO 14598 的内容基本协调。

我国在 2003 年颁布的 GB/T 16260—2003（ISO 9126—2001）《软件工程　产品质量》取代了 1996 年颁布的 GB/T 16260《软件产品评价—质量特性及其使用指南》。

2．标准之间的关系

GB/T 18905.1 概述是产品评价标准的总则，GB/T 16260 的评价过程与度量是遵循

GB/T 18905 的。如图 3-10 所示描述了 GB/T 18905 和 GB/T 16260 系列标准之间的关系。

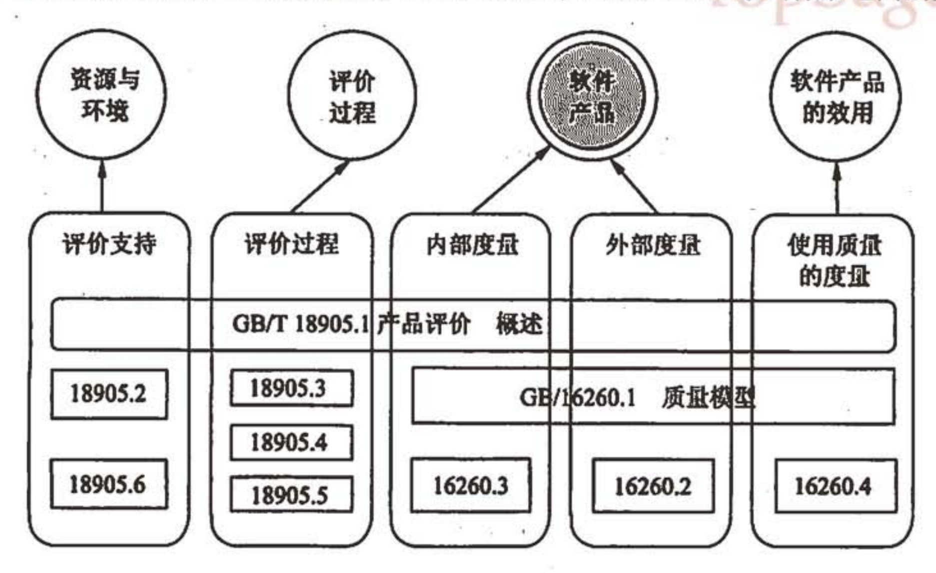


图 3-10 GB/T 18905 和 GB/T 16260 系列标准之间的关系

GB/T 18905 的各个部分应与 GB/T 16260 中描述软件质量特性和度量部分一起使用。

3.6.3 标准的范围

标准可从软件的获取、需求、开发、使用、评价、支持、维护、质量保证和审核相关的不同观点来确定和评价软件产品的质量。

可以被开发者、需方、质量保证人员和独立评价者，特别是那些对确定和评价软件产品质量负责的人员所使用。

本标准中定义的质量模型使用实例是：

- 确认需求定义的完整性；
- 确定软件需求；
- 确定软件设计目标；
- 确定软件测试目标；
- 确定质量保证准则；
- 为完整的软件产品确定验收准则。

软件质量标准还可以和软件质量保证过程，以及与软件过程改进有关的标准一起使用。

3.6.4　质量模型框架

1．软件质量特性与度量

- 质量特性和子特性（GB/T 16260.1）；
- 外部度量（GB/T 16260.2）；
- 内部度量（GB/T 16260.3）。

GB/T 16260.1 定义了质量特性、相关的子特性以及 GB/T 16260 质量模型上面三层之间的关系。GB/T 16260.2 和 GB/T 16260.3 确定了每种度量（外部和内部的）与其相应的特性和子特性之间的关系，如图 3-11 所示。注意，某些内部度量有对应的外部度量。

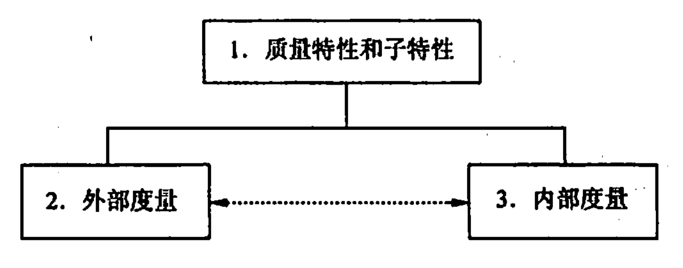


图 3-11　软件质量特性与度量

2．质量途径

新版 GB/T 16260 的用户质量要求包括：在特定的使用环境和条件下对使用质量的需求，而在规定外部质量和内部质量的特性和子特性时也可以使用这些需求。

软件产品质量可以通过测量内部属性，或者测量外部属性，或者测量使用质量的属性来评价。目标就是使产品在特定的使用环境和条件下具有所需的效用，如图 3-12 所示。

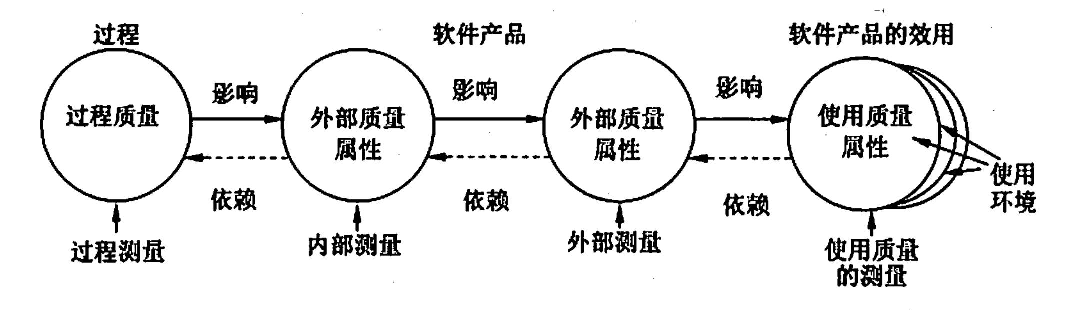


图 3-12　生存周期的质量途径

过程质量有助于提高产品质量，而提高产品质量有助于提高使用质量。因此，评估和改善过程是提高产品质量的一种手段，而评价和改进产品质量是提高使用质量的一种



方法。同样，评价使用质量可以为改进产品提供反馈，而评价产品则可以为改善过程提供反馈。适当的软件内部属性是获得所需外部特性的先决条件；而适当的外部特性则是获得使用质量的先决条件。

软件产品质量需求包括内部质量、外部质量和使用质量的评估准则，以满足开发者、维护者、需求方以及最终用户的需要。

3．产品质量和生存周期

内部质量、外部质量和使用质量的观点在软件生存周期中是变化的。

在软件生存周期的不同阶段存在着关于产品质量和相关度量的不同观点，如图 3-13 所示。

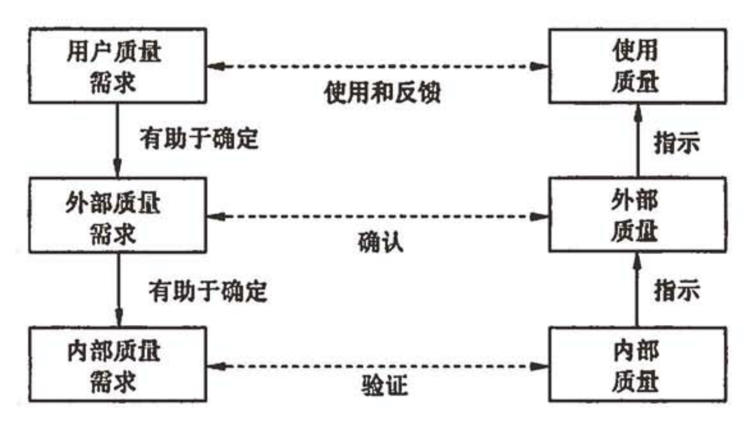


图 3-13 软件生存周期的质量

（1）用户的质量要求

可按使用质量的度量、外部度量或内部度量来规定质量需求。当验证产品时，这些由度量规定的需求应作为准则使用。

（2）外部质量需求

从外部观点来规定必须的质量级别，包括来源于用户质量要求（使用质量需求）。外部质量需求用作不同开发阶段的验证目标。

（3）内部质量需求

内部质量需求是从产品的内部观点来规定必须的质量水平。内部质量需求用来规定中间产品的属性，包括静态的和动态的模型、其他的文档和源代码。内部质量需求可用作不同开发阶段的验证目标。

（4）使用质量

使用质量是从用户观点出发，来看待软件产品用于特定环境和条件下的质量。它测

量用户在特定环境中达到其任务目标的程度，而不是测量软件自身的性质。

（5）外部质量

外部质量是从外部观点出发的软件产品特性的总体。它是当软件执行时，更典型地是使用外部度量在模拟环境中，用模拟数据测试时，所被测量和评价的质量。

（6）内部质量

内部质量是从内部观点出发的软件产品特性的总体。内部质量是针对内部质量需求被测量和评价的质量。

3.6.5　外部质量和内部质量的质量模型

外部质量和内部质量的质量模型软件，其质量属性划分为 6 种特性（功能性、可靠性、易用性、效率、维护性和可移植性），并进一步细分为一些子特性，如图 3-14 所示。

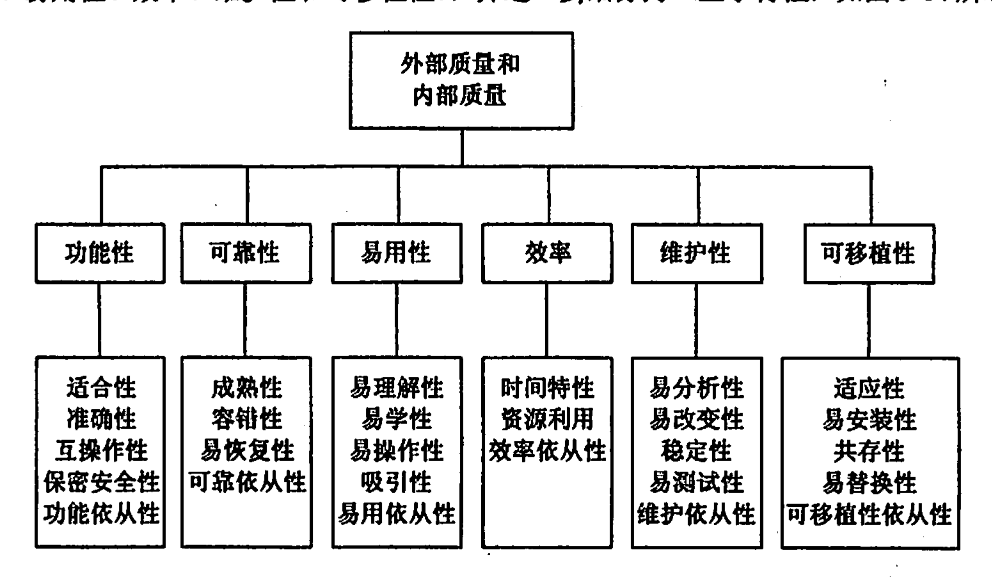


图 3-14　外部质量和内部质量的质量模型

软件的每个质量特性和子特性都有定义。对于每个特性和子特性，软件的能力由可测量的一组内部属性决定，内部度量的示例在 GB/T 16260.3 中给出。这些特性和子特性根据包含该软件的系统提供能力的程度从外部进行测量，外部度量的示例在 GB/T 16260.2 中给出。

1．功能性

功能性是指当软件在指定条件下使用时，软件产品满足明确和隐含要求功能的能力。

（1）适合性

适合性是指软件产品为指定的任务和用户目标提供一组合适的功能的能力。

（2）准确性

准确性是指软件产品具有所需精确度的正确或相符的结果及效果的能力。

（3）互操作性

互操作性是指软件产品与一个或更多的规定系统进行交互的能力。

（4）保密安全

保密安全是指软件产品保护信息和数据的能力，以使未授权的人员或系统不能阅读或修改这些信息和数据，但不拒绝授权人员或系统对它们的访问。

（5）功能性依从性

功能性依从性是指软件产品依附于同功能性相关的标准、约定或法规以及类似规定的能力。

2．可靠性

在指定条件下使用时，软件产品维持规定的性能级别的能力。

（1）成熟性

成熟性是指软件产品避免因软件中错误的发生而导致失效的能力。

（2）容错性

容错性是指在软件发生故障或者违反指定接口的情况下，软件产品维持规定的性能级别的能力。

（3）易恢复性

易恢复性是指在失效发生的情况下，软件产品重建规定的性能级别并恢复受直接影响的数据的能力。

（4）可靠性依从性

可靠性依从性是指软件产品依附于同可靠性相关的标准、约定或规定的能力。

3．易用性

易用性是指在指定条件下使用时，软件产品被理解、学习、使用和吸引用户的能力。

（1）易理解性

易理解性是指软件产品使用户能理解软件是否合适以及如何能将软件用于特定的任务和使用环境的能力。

（2）易学性

易学性是指软件产品使用户能学习它的能力。

（3）易操作性

易操作性是指软件产品使用户能操作和控制它的能力。

（4）吸引性

吸引性是指软件产品吸引用户的能力。

（5）易用性依从性

易用性依从性是指软件产品依附于同易用性相关的标准、约定、风格指南或规定的能力。

4．效率

效率是指在规定条件下，相对于所用资源的数量，软件产品可提供适当的性能的能力。

（1）时间特性

时间特性是指在规定条件下，软件产品执行其功能时，提供适当的响应和处理时间以及吞吐率的能力。

（2）资源利用性

资源利用性是指在规定条件下，软件产品执行其功能时，使用合适的数量和类型的资源的能力。

（3）效率依从性

效率依从性是指软件产品依附于同效率相关的标准或约定的能力。

5．维护性

维护性是指软件产品可被修改的能力。修改可能包括修正、改进或软件适应环境、需求和功能规格说明中的变化。

（1）易分析性

易分析性是指软件产品诊断软件中的缺陷或失效原因，以及判定待修改的部分的能力。

（2）易改变性

易改变性是指软件产品使指定的修改可以被实现的能力。

（3）稳定性

稳定性是指软件产品避免由于软件修改而造成意外结果的能力。

（4）易测试性

易测试性是指软件产品使已修改软件能被确认的能力。

（5）维护性依从性

维护性依从性是指软件产品依附于同维护性相关的标准或约定的能力。

6．可移植性

可移植性是指软件产品从一种环境迁移到另外一种环境的能力。

（1）适应性

适应性是指软件产品无需采用有别于为考虑该软件的目的而准备的活动或手段，就可能适应不同的指定环境的能力。

（2）易安装性

易安装性是指软件产品在指定环境中被安装的能力。

（3）共存性

共存性是指软件产品在公共环境中同与其分享公共资源的其他独立软件共存的能力。

（4）易替换性

易替换性是指软件产品在环境相同、目的相同的情况下替代另一个指定软件产品的能力。

（5）可移植性依从性

可移植性依从性是指软件产品依附于同可移植性相关的标准或约定的能力。

3.6.6 使用质量的质量模型

使用质量是从用户角度看待的质量,其属性分为 4 种：有效性、生产率、安全性和满意度，如图 3-15 所示。

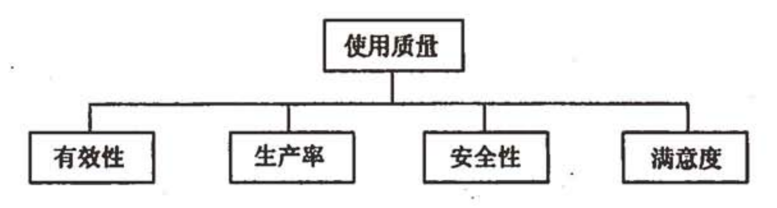


图 3-15 使用质量的质量模型

软件产品具有指定用户在指定的使用环境下，获得与有效性、生产率、安全性和满意度相关的规定目标的能力。

（1）有效性

有效性是指软件产品在指定的使用环境下，使用户获得满足准确度和完整性要求的规定目标的能力。

（2）生产率

生产率是指软件产品在指定的使用环境下，使用户可使用与获得的有效性有关的合适数量资源的能力。

（3）安全性

安全性是指软件产品在指定使用环境下，获得可接受的对人类、事务、软件、财产

或环境有害的风险级别的能力。

（4）满意度

满意度是指软件产品在指定使用环境下，使用户满意的能力。

3.7　软件测试国家标准

目前广泛使用的软件测试国家标准包括：

- GB/T 9386－1988《计算机软件测试文件编制规范》。
- GB/T 15532－1995《计算机软件单元测试规范》。
- GB/T 17544－1998《信息技术 软件包 质量要求和测试》。
- GB/T 16260.1－2003《软件工程　产品质量》第 1 部分，质量模型。
- GB/T 16260.2－200X《软件工程　产品质量》第 2 部分，外部度量。
- GB/T 16260.3－200X《软件工程　产品质量》第 3 部分，内部度量。
- GB/T 16260.4－200X《软件工程　产品质量》第 4 部分，使用质量度量。
- GB/T 18905.1－2002《软件工程　产品质量》第 1 部分，概述。
- GB/T 18905.2－2002《软件工程　产品质量》第 2 部分，策划和管理。
- GB/T 18905.3－2002《软件工程　产品质量》第 3 部分，开发者用的过程。
- GB/T 18905.4－2002《软件工程　产品质量》第 4 部分，需方用的过程。
- GB/T 18905.5－2002《软件工程　产品质量》第 5 部分，评价者用的过程。
- GB/T 18905.6－2002《软件工程　产品质量》第 6 部分，评价模块文档编制。

第 4 章　软件测试过程与管理

众所周知，采用先进的标准、方法和工具对于软件测试是十分重要的，但是成功的软件测试是离不开对测试的组织与过程的管理的，没有目标、没有组织、没有过程控制的测试是注定要失败的。一个软件的测试工作，不是一次简单的测试活动，它与软件开发一样，是属于软件工程中的一个项目，因此，软件测试的过程管理是测试成功的重要保证。

4.1　软件测试过程

开发过程的质量决定了软件的质量，同样地，测试过程的质量决定了软件测试的质量和有效性。软件测试过程的管理是保证测试过程质量、控制测试风险的重要活动。软件测试和软件开发一样，都遵循软件工程的原理，有它自己的生命周期。软件的测试过程一般分成测试计划、测试设计与开发、测试实施、测试评审与测试结论等阶段。对每个阶段的任务、输入和输出都有明确的规定，以便对整个测试过程进行质量控制和配置管理。

软件测试过程是一种抽象的、遵循 GB/T 18905（ISO 14598.5）《评价者用的过程（Process for Evaluator）》中定义软件评价过程的模型，是国际上共同遵守的软件评测过程标准，是软件测试过程管理的精髓。标准定义了分析各类软件产品的评测需求，规定、设计、实施、评审以及对评测做出结论所需的各种活动。本章介绍的主要内容，可作为软件测试过程工作内容与管理的基本原则。为符合 GB/T 18905 基本原理，仍保留“评价过程”的标准用语。

4.2　评价过程的特性

① 可重复性：由同一评价者按同一评价规格说明对同一产品进行重复地评价，应产生同一种可接受的结果。

② 可再现性：由不同评价者按同一评价规格说明对同一产品进行评价，应产生同一种可接受的结果。

③ 公正性：评价应不偏向任何特殊的结果。

④ 客观性：评价结果应是客观事实，即不带有评价者的感情色彩或主观意见。

4.3 评价过程

4.3.1 评价活动

评价过程由下列 5 个活动组成：

（1）确立软件评价需求

（2）编制评价规格说明

根据请求者提供的评价需求和产品描述编制。

（3）制定评价计划

在评价规格说明的基础上设计评价，需考虑要测软件的部件和评价者建议的评价方法。

（4）评价执行计划

- 按照评价计划对产品及其部件进行检查、建模、测量和评价。
- 可以用软件工具（通常由评价者提供）来实施。
- 记录评价者的执行动作，所得的结果被记入评价报告草案。

（5）作评价结论

交付评价报告和评价者对评价产品所做的处理。

4.3.2 评价过程的输入

请求者提供其需求，并作为评价需求的最初版本。

- 请求者提供下列评价过程的输入。

① 软件的说明书；

② 软件的部件。软件的说明书标识的软件产品以及供评价的部件。

- 评价者提供下列评价过程输入。

① 预先确定的评价规格说明；

② 评价方法；

③ 评价工具。

4.3.3 评价过程的输出

在评价期间，评价者提供下列输出产品：



① 评价记录，包括评价计划和评价动作的记录；

② 评价报告草案，包括评价需求，评价规格说明和综合的评价结果；

③ 经过评审的评价报告。

4.3.4 评价过程文档

评价需求、评价规格说明和评价计划是评价过程的中间产品；评价记录和评价报告是评价过程的最终产品。

① 评价需求：描述评价的目标，特别是描述了产品的质量需求。

② 评价规格说明：确定对软件及其部件实行的所有分析和测量，标识要分析和测量的软件部件。

③ 评价计划：描述评价规格，说明需要实施的操作规程；描述评价所需用到的方法和工具。

④ 评价记录：评价执行计划时详细记载的动作组成；记录由评价者保留。

⑤ 评价报告：执行测量和分析的结果，以及能被重复和重新评价的必要信息。评价报告首先作为评审草案来发布，其最终版本将交给请求者。

如图 4-1 所示给出了上面描述过程的概述，标识了各活动之间的信息流。

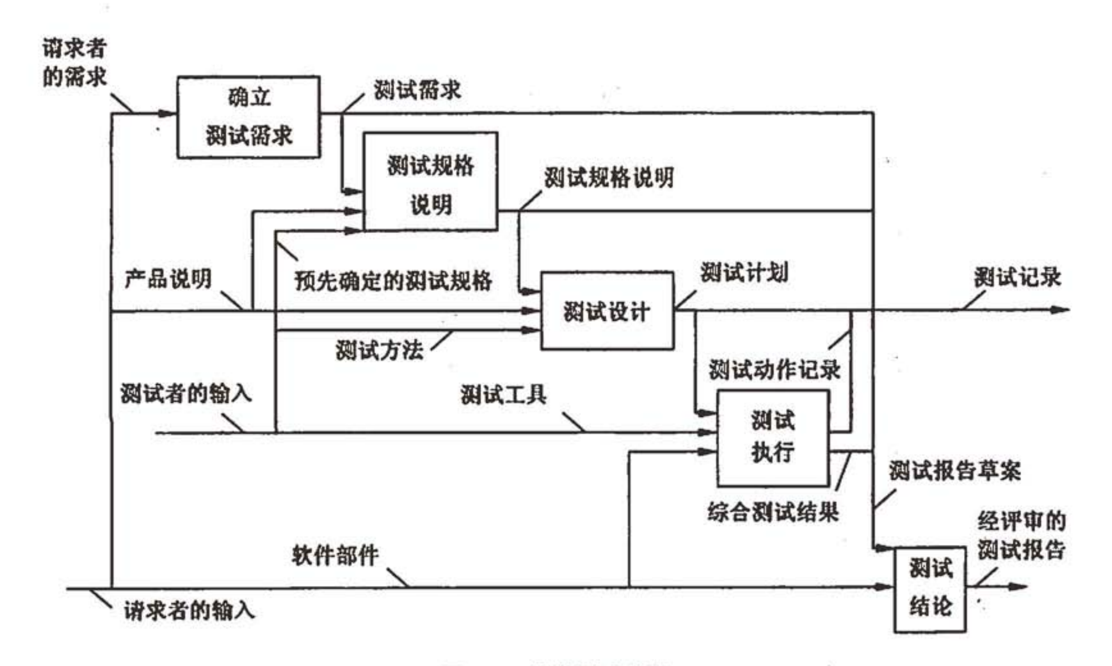


图 4-1 软件测试过程

4.4　评价与生存周期的关系

评价软件产品可以在任何软件生存周期过程的范围内进行。特别是，评价能在软件获取、供应、开发、运行或维护过程中进行。

在软件开发过程中可尽早决定是否执行软件产品的评价。如果在开发过程的开始阶段能确定下来，就有可能把评价要执行的测量和评价放入软件开发的过程中。这样能保证软件最大可能地满足有关评价结果的所有需求，降低额外风险和未预料的成本。当请求者也是软件的开发者时，及早与评价者联系，讨论将提交一个产品用于评价，会有助于开发者预见评价者可能提出的任何特殊要求。

可能有某些（或全部）评价动作必须在现场实施，而不是在评价者所在地完成。在这种情况下，为保证结果是公正的，这些动作仍受评价者的控制。

对于大而复杂的软件项目开发者来说，在整个产品的开发期间，开发者与评价者不断密切合作是很有益的，这将减少评价过程的成本和时间。但这种合作应不降低评价者的公正性。

4.5　评价过程的要求

4.5.1　一般要求

1．组织和质量体系

为了满足评价结果的可重复性、可再现性、公正性及客观性，评价者应立足于一个组织。该组织为使其活动达到充分的质量要求提供所有必要保证。为满足这一需求，评价组织可以按照 ISO/IEC 17025 中规定的要求建立质量管理体系。

2．请求者的职责

评价请求者的职责应包括：

① 为进行评价，对软件产品确立必要的合法权利；

② 为标识和描述产品提供必要的信息；

③ 阐述最初的评价需求，并与评价者协商，确定实际的评价需求，这些评价需求宜遵守相关的法规和标准；

④ 阐述对评价提交的信息的保密性需求；

⑤ 必要时在开发者和评价者之间起中介作用；

⑥ 必要时向评价者提供对用于开发和操作使用软件产品的计算机和其他设备；



⑦ 必要时对评价者提供必要的支持，包括培训和走访；

⑧ 必要时确保及时提供软件、产品说明书和部件，包括文档及其他资料；

⑨ 必要时告知评价者可能导致评价结果无效的原因。

3．评价者的职责

评价者的职责应是：

① 检查请求者对要评价执行的软件产品是否有充分合法的权利；

② 按规定对请求者提供信息保密承诺，包括评价的软件、评价记录和评价报告；

③ 提供有资格的和经过培训的人员，以便实施评价；

④ 提供评价工具和技术；

⑤ 按照评价需求实施测试；

⑥ 保留评价期间影响评价结果的所有工作记录；

⑦ 保证及时向请求者提交评价报告。

4.5.2 评价需求确立

1．评价需求确立的目的

评价需求确立的目的是描述评价目标。这些目标关系到软件产品的预期用途和相关风险。可能要从几种不同软件用户的角度出发，如软件的需方、供方、开发者、操作者或维护者。

2．评价需求分析

分析评价需求的活动由下列5个子活动组成：

① 请求者提出评价需求建议；

② 请求者说明评价覆盖范围；

③ 评价者分析评价原因和描述评价需求来响应请求者；

④ 评价者解释评价的保密范围和严格程度；

⑤ 评价者同意评价需求。

进行评价需求分析时，要考虑供评价的产品的应用领域和用途，还要考虑一些关键问题，如安全、保密安全、经济或环境方面的问题，以及适用的法律和规章制度。

在请求者的需求中，请求者应表明评价覆盖范围。同时评价者应保证评价是非常严格的，足以提供软件产品质量方面的真实证据。因此，请求者与评价者应对评价需求达成一致，作为继续评价过程的前提条件。

3．评价需求内容

① 评价需求应包含对评价产品应用领域的描述，以及产品用途的描述。

② 评价需求应由GB/T 16260中定义为“质量特性”的一系列质量需求组成，还可

能用到一些子特性。

③ 评价需求中的每项需求，都应提供要评价软件及部件的规格说明信息。

4．认可与报告

① 评价需求应作为请求者与评价者联合评审的结果而予以承认；

② 评价需求应包括在评价报告和评价记录中。

4.5.3 评价规格说明

1．评价规格说明的目的

规定评价规格说明的目的是定义评价范围，定义供评价产品及各种部件执行的测量。评价规格说明应详细到以此能确保评价的可重复性和可再现性。

2．评价规格说明编制

编制评价规格说明的活动由下列 3 个子活动组成：

① 分析产品的描述；

② 规定对产品及部件执行的测量；

③ 按照评价需求验证编制的规格说明。

（1）产品说明分析

请求者应提供提交要测软件的产品说明，目的是：

① 以此定义评价的范围，即标识出哪些作为软件一部分的部件，以及便于了解而接触软件的情况；

② 将供评价的产品部件的标识交给评价者，以便评价者了解其结构，并弄清提供的信息及如何访问产品部件。

产品说明资料应包含为评价而实际提交的产品部件清单、有关产品结构的基本原理和与产品有关的文档清单。对列在清单中的每个部件和与产品有关的文档，应提供下列信息：

① 部件性质的描述；

② 部件中用到的形式化信息；

③ 有关部件规模的信息；

④ 与其他部件的关系；

⑤ 对评价者有用的产品部件信息。

评价者应检查产品描述是否与上述提及的需求一致。评价者还应分析提供的原理及部件的说明，以便标识在评价需求中确定的各部件间的关系。

（2）测量规定

评价者应把评价需求分配给产品本身和产品描述中标识的各种部件，使评价需求被

分解为数个子特性。对供测试的不同部件，分解的结果也是不同的。然后，测试者应规定旨在对产品和所选部件的特性、子特性及属性进行评估的测量。测试规格说明书应明确对下列几项进行说明：

① 用于度量软件或一组标识的部件的形式化的规格说明，以及评价报告中测量结果的表现形式的说明；

② 引用的产品部件中规定将要验证的软件需求，以及引用验证这些需求的规程的说明；

③ 在软件需求文档中被遗漏的，或需要更详细解释的软件产品的需求规格说明，以及用来验证这一需求的规程的说明。

对上述这些声明，应引用要测量或验证的部件的性质和所用的形式。

（3）评价规格说明验证

评价者应按照评价需求来验证评价规格说明。

评价者应按照评价需求检查列在产品描述中的部件是否提供了评价执行的所有必要信息。评价者还应验证规定的测量和验证是否充分满足了评价需求所表示的评价目标。

3．评价规格说明的内容

评价规格说明应包括：

① 评价范围，涉及在产品说明中标识的产品部件；

② 评价执行所需的信息，在产品说明中列出的软件部件及其他相关文档之间的相互引用；

③ 要执行的测量和验证的规格说明，以及对要评价的产品部件的引用；

④ 测量和验证的规格说明与评价需求之间，与引用标准或对所列的每个测量或验证的理由之间的映射。

4．认可和报告

评价规格说明应作为请求者和测试者之间联合评审的结果予以认可。

评价规格说明应包含在评价报告和评价记录中。此外，对评价需求的任何修改均应在评价记录中予以报告。

4.5.4 评价设计

1．评价设计目的

评价设计应把评价者使用的测量规程编成文档，以便评价执行规格说明中规定的测量。评价者应制定评价计划来描述执行指定的评价时所需的资源和执行各种动作时对这些资源的分配。

评价计划应详细到能确保用一种令人满意的方式执行这些动作。

2．制定评价计划

制定评价计划的活动由 3 个子活动组成：

① 把评价方法编成文档，起草计划；

② 优化评价计划；

③ 根据可用资源安排评价动作的进度。

（1）编制评价方法文档和起草计划

把规定的测量或验证与要评价的各种产品部件的形式组合起来，以便把对部件实施的测量或验证的详细方法编成文档。

评价者应分析评价规格说明中规定的有关测量或验证的技术约束条件。这些约束条件可能包括：

① 软件部件所用的形式；

② 软件部件说明的电子或书面形式；

③ 预定义评价方法；

④ 支持评价技术的工具的可用性；

⑤ 软件部件的规模。

对评价规格说明中规定的每种测量或验证，评价者都应把有关的评价方法编成文档。

当描述的评价方法是基于使用软件工具的时候，应在评价计划中标识该工具。这种标识应至少包括工具的名称、版本标识和它的来源（如：供方）。

对评价的产品执行程序时，也应说明运行环境以及实际工作中可用的条款。

（2）测量的优化

每个基本评价方法都计划应用在供评价的各个软件部件上。也会出现将不同的基本评价方法用于同一个软件部件的情况。

应对评价计划草案进行评审，以避免评价者的重复劳动，减少错误风险和降低计划的评价者的工作量。

（3）安排评价动作的进度

评价者应安排计划动作的进度，评价者应考虑人员、软件工具、计算机等资源的可用性。

评价者应就软件及部件的交付进度与请求者达成一致。应规定软件部件的交付介质、形式以及拷贝数量。

应标识评价过程中满足的需求。当请求者不是软件产品的开发者时，应标识评价者和开发者之间的关系。特别是应规定开发者需要的支持。这种支持包括培训、非正式的



讨论或办公场所。

必要时，对开发和运行场所的访问也应与所需资源一起规定。

（4）评价计划的内容

评价计划应由两部分组成：评价方法文档和评价者采取评价动作的时间表。

评价计划中某些评价方法的文档可能包括对评价者个人材料的引用。在这种情况下，评价者应能判断该方法与相应评价规格说明元素的针对性，以及在应用该方法时，其自身的能力。

3．认可和报告

评价计划应作为请求者和评价者之间联合评审的结果而予以认可。

评价计划应包含在评价记录中。评价方法的文档，对方法的引用，以及对要应用该评价方法的产品部件的标识都应在评价报告中体现。

4.5.5 评价执行

1．评价执行目的

评价执行目的是根据评价需求，按照评价规格说明中的规定和评价计划，从对软件产品的测量和验证中获得结果，执行这些动作将完成评价报告和评价记录的草稿。

2．评价执行者的动作

为了执行计划的评价，评价者应做到以下几点。

① 管理请求者提供的产品部件；

② 管理评价动作所产生的数据（包括报告和记录）；

③ 管理评价执行动作的工具。

此外，评价者还可以管理在评价者的承诺之外执行的评价动作；

管理使用特定评价技术所隐含的要求。

（1）软件部件的管理

评价请求者应根据评价计划中定义的进度向评价者交付软件部件和与软件相关的文档。

评价者应登记全部软件部件和软件的相关文档。在证实了软件的规模和复杂程度之后，应使用正式的配置管理。

软件样品登记的信息应至少包括：

① 部件或文档的惟一标识符；

② 部件的名称或文档标题；

③ 文档的状态（包括物理状态或变异状态）；

④ 请求者提供样品的版本、配置和日期信息；

⑤ 接收的日期。

除非请求者有另外的许可，否则，评价者将保守全部产品部件和相关文档的秘密。

（2）评价数据管理

评价执行动作通常是测量产品和它的部件，以获得并解释中间数据，以便将产生的结果记入测试报告。中间数据的种类多种多样，例如，为测试产生的数字、图形、图表、部件的摘录或形式化模型。

对中间数据的保密应与原来对部件和文档的保密方法一样。此外，评价者应尽力防止这些数据被意外或恶意地修改。特别是当中间数据量非常大并且非常复杂时，应使用正式的配置管理来保持中间评价结果与评价产品之间的一致性。

评价者应把所有中间数据记入评价记录，以便依据它们进行解释。如同在评价计划中规定的那样，在解释过程中所作的决定也应被记入评价记录。

（3）工具使用的管理

评价执行动作需要使用软件工具来收集原始数据，或解释中间数据。

当使用工具来评价执行动作时，应在评价报告中记录对工具的引用。该引用应由工具的标识、工具的供方和工具的版本信息组成。

对所用工具的更详细的引用信息应记录在评价记录中，包括工具配置的详细信息和为得到相同的中间结果而重复评价动作所需的任何相关信息。

评价者应尽最大努力保证工具按照所期望的方式进行工作。评价者应保留在评价过程中承诺合法使用工具的记录。

（4）现场评价

有时，不能在评价者假定的场所评价执行动作。如，开发者的工作地点或软件产品的运行现场。

这时，评价者应控制所有执行的评价动作。特别是，应避免任何使评价结果无效的情况发生。

评价者应尽最大努力保证评价结果和中间结果的保密性。

（5）特定评价技术的需求

当评价计划要求评价产品的可执行程序时，应精确记录评价的配置和评价的环境。

当评价动作要求检查文档时，建议使用检查表。

（6）评审和报告

在评价执行过程中会产生中间评价结果和最终评价结果。为达到最大的客观性，每个评价动作应由不同的评价执行动作的评价者来检查。



应评审全部评价结果，其目的取决于所考虑的评价动作的实质。应至少有一个不直接涉及评价动作的人员参加评审。评审报告应包括在评价记录中。

一旦评审通过，应像评价规格说明中规定的那样，把评价结果记入评价报告中。此外，当评价计划也是这样规定时，某些中间结果或解释决定也应记入评价报告。

4.5.6 评价结论

1．评价结论的目的

评价结论的目的包括评价报告的评审和评价数据的处理。

2．评价报告的联合评审

评价报告的草稿应交付评价的请求者。应组织评价者和请求者之间的联合评审。请求者应有机会对评价报告提出意见。之后，应把该评价报告交给请求者。

3．评价数据和文档的处置

将评价报告正式交付给请求者之后，评价者应处理与评价有关的数据。

可以根据数据的类型使用下列方法进行：

① 供评价的文档应归还给请求者，或者存档一个规定的期限，或者以安全的方式销毁；

② 评价报告和评价记录应存档一个规定的期限；

③ 所有其他数据应存档一个规定的期限或以安全的方式销毁。

当某些数据的规定存档期限到期时，应将其再次保存一个规定的期限或以安全的方式销毁。

只要请求者明确表示同意，评价者就可以使用中间评价结果，以便研究评价技术和软件的度量。

4.6 配置管理

软件测试过程的配置管理和软件开发过程的配置管理是一样的。开发过程中，测试活动的配置管理属于整个软件项目配置管理的一部分。独立的测试组织应建立专门的配置管理系统。一般来说，软件测试配置管理包括 4 个最基本的活动：

① 配置项标识；

② 配置项控制（变更控制）；

③ 配置状态报告；

④ 配置审计。

4.6.1 配置项标识

- 标识测试样品、标准、工具、文档（包括测试用例）、报告等配置项的名称和类型。
- 指出何时基准化配置项（置于基线控制之下）。
- 标识各配置项的所有者及储存位置。

4.6.2 配置项控制

- 规定测试基线，对每个基线必须描述下列内容：

① 每个基线的项（包括文档、样品和工具）；

② 与每个基线有关的评审、批准事项以及验收标准。

- 规定何时何人创立新的基线，如何创立。
- 确定变更控制委员会的人员组成、职能（包括变更授权、确认与批准）、工作程序。
- 确定变更请求的处理程序和终止条件。
- 确定变更请求的处理过程中各测试人员执行变更的职能。
- 确定变更请求和所产生结果的对应机制。
- 确定配置项提取和存入的控制机制与方式。

4.6.3 配置状态报告

- 定义配置状态报告形式、内容和提交方式。
- 确认过程记录和跟踪问题报告，更改请求，更改次序等。
- 确定测试报告提交的时间与方式。

4.6.4 配置审计

- 确定审计执行人员和执行时机。
- 确定审计的内容与方式。
- 确定发现问题的处理方法。

配置管理是管理和调整变更（change）的关键（如图 4-2所示），对于一个参与人员较多、变更较大的项目，它是至关重要的。软件测试配置管理概念相对比较简单，但实际操作却常常十分复杂。它应用于测试工具、用例，而且对于测试过程中的所有文档也是非常重要的，也可应用于测试样本和数据。

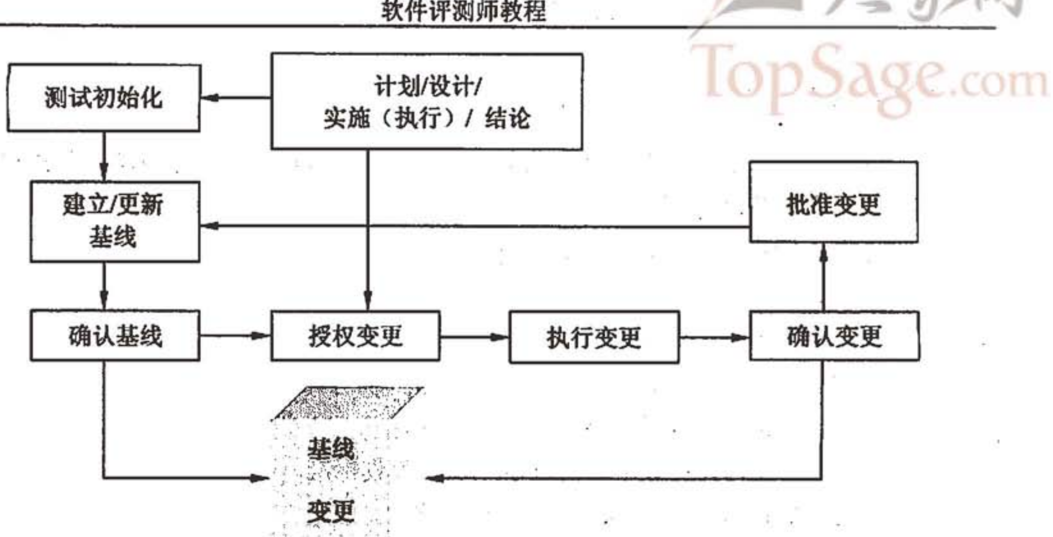


图 4-2 配置管理原理图

4.7 测试的组织与人员

组织是指一个系统将材料、知识和方法组合起来，把各种不同的输入转换成有价值的输出。组织结构是指用一定的模式对责任、权威和关系进行安排，直至通过这种结构发挥功能。

4.7.1 组织结构设计因素

测试组织结构设计因素包括：

- 垂直还是平级：垂直的组织结构是在首席管理者与低级测试人员之间设立许多层次，平级的垂直组织结构设立很少的几个层次。平级的组织结构的测试工作效率较高。
- 市场还是产品：组织结构的设置可以是面向不同的市场或不同的产品。
- 集中还是分散：组织可以是集中的，也可以是分散的。这对于测试组织是比较关键的，为保证测试的独立性，一般测试组织要相对集中。
- 分级还是分散：可以将组织按权力和级别一层一层地分级。也可以分散排列开。在软件开发小组内的测试常使用这种分散的方式，测试小组在开发小组内，可以是专职测试人员，或者以测试角色的形式组成。
- 专业人员还是工作人员：测试组织应拥有一定比例的专业测试人员和工作人员。

- 功能还是项目：测试组织可以面向功能或项目。

4.7.2 独立测试组织

测试组织是一种资源或一系列的资源，专门从事测试活动。随着软件企业规模的不断增大，必须建立独立专门的测试队伍。只有不持偏见的人才能提供不持偏见的度量，测试度量软件质量才真正有效，测试必须独立进行。

Bill Hetzel 在《软件测试指南大全》（1988）一书写道："独立的测试组织十分重要，因为，①没有这样的一个组织，建立系统就不会理想；②有效的度量对于产品质量控制是十分重要的；③测试协调需要全职、专门的人员投入"。

4.7.3 测试组织管理者

测试管理是很困难的，测试组织的管理者必须具备：

- 理解与评价软件测试政策、标准、过程、工具、培训和度量的能力；
- 领导一个测试组织的能力，该组织必须坚强有力、独立自主、办事规范没有偏见；
- 吸引并留住杰出测试专业人才的能力；
- 领导、沟通、支持和控制的能力；
- 测试时间、质量和成本控制的能力。

4.7.4 集中管理的测试组织

将 4.1 节中组织的基本设计因素组合起来，可以构成许多不同的测试组织结构。本节重点介绍在软件企业中，与独立测试有关的集中管理的一种测试组织形式（如图 4-3 所示）。

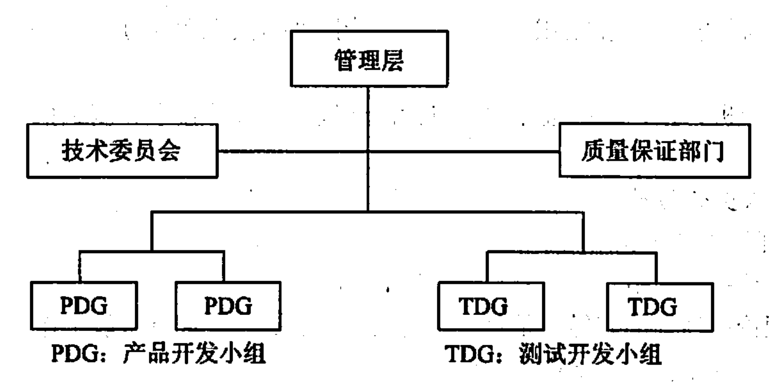


图 4-3　集中管理的测试组织



这种方式的优点是，软件立项后，由独立的测试组织提供资源与软件开发人员并肩作战，与合作伙伴一块行动，可以减少软件开发人员与测试人员合作时的不利因素。

4.7.5 选择合理的组织方案

组织设计因素可以组成不同的组织方案，在实际中软件开发机构和测试机构也都建立了不同结构的测试组织形式。选择合理高效的测试组织结构方案的准则是：

① 提供软件测试的快速决策能力；

② 利于合作，尤其是产品开发与测试开发之间的合作；

③ 能够独立、规范、不带偏见地运作并具有精干的人员配置；

④ 有利于协调测试与质量管理的关系；

⑤ 有利于满足软件测试过程管理要求；

⑥ 有利于为测试技术提供专有技术；

⑦ 充分利用现有测试资源，特别是人；

⑧ 对测试者的职业道德和事业产生积极的影响。

4.7.6 测试人员

1．测试人员的选择

测试人员的能力包括以下几项。

① 一般能力：包括表达、交流、协调、管理、质量意识、过程方法、软件工程等；

② 测试技能及方法：包括测试基本概念及方法、测试工具及环境、专业测试标准、工作成绩评估等；

③ 测试规划能力：包括风险分析及防范、软件放行/接收准则制定、测试目标及计划、测试计划和设计的评审方法等；

④ 测试执行能力：包括测试数据/脚本/用例、测试比较及分析、缺陷记录及处理、自动化工具；

⑤ 测试分析、报告和改进能力：包括测试度量、统计技术、测试报告、过程监测及持续改进。

2．测试人员的激励

（1）X 理论 + Y 理论

- X 理论：胡萝卜 + 大棒——迫使人们工作；
- Y 理论：经理的职能不是督促人们工作，而是使人们有可能工作。

（2）需要的层次（Maslow 模型）

- 生存需要——工作职位、工资奖金、休息时间；

- 安全需要——公正待遇、应付工作的能力和信心；
- 社会需要——团队归属感，互相认同、理解和支持；
- 自尊需要——具有受人尊重/赏识的能力或/和业绩；
- 自我实现需要——成为自己期望的人物。

（3）人员激励的关键点

- 管理者习惯用对自己有效的因素激励测试人员，很可能发现无效；
- 过多使用权力、资金或处罚手段很可能导致项目失败；
- 行业领先企业采取卓有成效的非货币形式的激励措施；
- 在项目进行过程中，而不仅是在项目结束时实施激励措施；
- 奖励应该在工作获得认同后尽快兑现；
- 对人员的工作表现出真诚的兴趣是对他们最好的奖励；
- 激励因素是因人而异、因时而异的。已经满足的需要很可能不再成为激励因素。

（4）人员自我激励

测试工作的快乐哲学：选择积极的态度，把工作当作游戏，让别人快乐，全身心投入工作。

注意测试工作的 7 条效率原则：主动思考，积极行动；一开始就牢记目标，不迷失方向；重要的事情放在首位（但常常把紧急的事情放在首位）；先理解人，后被人理解；寻求双赢；互相合作，追求 1+1>2；终生学习，自我更新，不断进步。

3．测试职业发展

国际推荐的软件测试职业发展计划如下。

- 1～2 年，测试技能：熟悉整个测试过程及产品业务领域，学习和掌握自动测试工具，学习测试自动化编程技术；开发和执行测试脚本，承担系统测试实施任务；掌握编程语言、操作系统、网络与数据库方面的技能。
- 3～4 年，测试过程：深入了解测试过程，掌握测试过程设计及改进，参与软件工作产品的同行评审；进一步了解产品业务领域，改进测试自动化编程技术；能指导初级测试工程师；加强编程语言、操作系统、网络与数据库方面的技能。
- 4～5 年，测试组织工作：管理 1～3 名测试工程师，担任任务估算、管理及进度控制；进一步培养在软件项目管理及支持工具方面的技能。
- 5～6 年，技术管理：管理 4～8 名测试工程师，提高任务估算、管理及进度控制能力，完成测试规划并制定测试计划；研究测试的技术手段，保持使用项目管理及支持工具的技能；用大量时间为其他测试工程师提供技术及过程方面的指导；开始与客户打交道并做演示推介。
- 6～12 年，测试管理：管理 8 名以上测试工程师，负责一个或多个项目的测试



工作；与客户打交道并做演示推介；保持使用项目管理及支持工具的技能。

4．人员的培训

（1）软件测试培训内容分类

- 测试基础知识和技能培训。
- 测试设计培训、测试工具培训。
- 测试对象——软件产品培训。
- 测试过程培训。
- 测试管理培训。

（2）制定测试人员培训计划

- 是测试计划的一个重要组成部分。
- 需要管理层的重视，在时间和资源上予以保证。
- 认真调查和分析测试人员的培训需求。
- 将培训活动安排在测试任务开始前。
- “边干边学”模式很可能牺牲质量和效率。
- 软件测试实习活动在整个培训中占较大比例。
- 鼓励合作学习，团队演练。
- 对培训效果要及时评价，发现不足进行改进。

4.8 软件测试风险分析

4.8.1 软件测试与商业风险

软件公司的管理者制定整体软件开发战略时使用“计划—执行—检查—改进（PDCA）”循环理念，战略性的策略可以转为商业上的主动。在PDCA循环中，计划（plan）和执行（do）往往更被人们所重视，然而，检测（check）部分才是用来处理商业风险的关键过程。

软件测试是一种用来尽可能降低软件风险的控制措施。软件测试是检测软件开发是否符合计划，是否达到预期的结果的测试。如果检测（check）表明软件的实现没有按照计划执行或与预期目标不符，就要采取必要的改进行动（active）。因此，公司的管理者应该依靠软件测试之类的控制措施来帮助自己实现商业目标。

软件测试人员必须明白他们的任务之一就是通过测试来评估产品的商业风险，并将结果报告给公司管理者。从这个角度看来，测试人员首先要理解什么是商业风险，并且要以这些风险为重点来制定测试策略。

4.8.2　什么是软件风险

风险的定义为“伤害、损坏或损失的可能性；一种危险的可能或一种冒险事件。”风险涉及到一个事件发生的可能性，涉及到该事件产生的不良后果或影响。软件风险是指开发不成功引起损失的可能性，这种不成功事件会导致公司商业上的失败。

软件测试中的风险分析是根据预测软件将出现的风险，制定软件测试计划并排列优先等级。风险分析是对软件中潜在的问题进行识别、估计和评价的过程。

软件风险分析的目的是确定测试对象、测试优先级以及测试的深度。有时还包括确定可以忽略的测试对象。通过风险分析，测试人员识别软件中高风险的部分并进行严格彻底的测试；确定潜在的隐患软件构件，对其进行重点测试。在制定测试计划的过程中，可以将风险分析的结果用来确定软件测试的优先级与测试深度。

软件风险分析工作应由各部门的专家组成，一般包括：项目经理、开发人员、测试人员、用户、客户以及销售人员。

对所有的软件项目进行风险分析将是必不可少的。如果软件本身的缺陷与错误能够导致灾难性后果，如造成严重的经济损失或生命危害，这样的软件称为安全性重要软件，安全性重要软件在开发过程中的各个阶段都应进行安全性分析。

即使是非重要软件，在项目的初期进行风险分析，也有助于识别潜在的问题。这些问题可能会引发严重的后果，因此项目经理和开发人员在开发中要特别注意，以便预防风险。

测试人员可利用风险分析的结果选择最关键的测试，大部分的测试资源应该用在控制最高级别的商业风险上，而最低级别的商业风险应该占用尽可能少的测试资源。只有这样，软件测试人员才能制定合理的策略，控制软件开发的风险。

4.8.3　软件风险分析

风险分析是一个对潜在问题识别和评估的过程，即对测试的对象进行优先级划分。风险分析包括两个部分：

① 发生的可能性—发生问题的可能性有多大。

② 影响严重性—如果问题发生了会有什么后果。

风险分析由以下几个步骤组成：首先列出潜在问题，然后对标识的每个潜在问题发生的可能性和影响严重性赋值，进行风险测定。测试人员根据测试分析结果的排列，关注潜在问题，设计与选择测试用例。

通常风险分析采用两种方法：表格分析法和矩阵分析法。通用的风险分析表包括以下几项内容。



① 风险标识（ID）——表示风险事件的惟一标识。
② 风险问题——问题发生现象的简要描述。
③ 发生的可能性——可能性值从1（低）～10（高）。
④ 影响的严重性——严重性值从1（低）～10（高）。
⑤ 风险预测值——发生可能性和影响严重性的乘积。
⑥ 风险优先级——风险预测值从高到低的排序。

软件风险分析表的例子如表4-1所示。

表4-1 软件风险分析表

标识	风险问题	可能性	严重性	预测值	优先级	测试用例
A	非法用户访问	6	8	48	2	TC-1-1
B	非法数据输入	7	10	70	1	TC-1-2
C	数据库更新不同步	4	10	40	4	TC-2-1
D	并发用户少	5	9	45	3	TC-3-1
E	用户文档不清晰	9	1	9	5	TC-4-1

可能性与严重性的乘积产生的风险预测值，决定了风险优先级的排序。预测值越高，优先级别越高，针对该问题的测试就越重要。根据表4-1的计算结果风险问题的排列为B、A、D、C、E。在风险计算过程中，可能出现具有相同预测值的情况，有的测试机构可以通过将可能性和严重性分别加权计算来进行进一步的分析。

本书风险分析的可能性值和严重性值的范围推荐使用从1～10，有些机构可能使用值的范围为1～100，或0～1之间的小数，也有的机构使用高、中、低三个等级来表示。至于使用哪种等级表示并不是很重要，只要这些值在分析过程中的使用是一致的，分析的效果都是一样的。

风险矩阵是风险分析的另一种有效的方法，测试人员可根据需要对风险潜在问题的可能性和严重性采用高（1）、中（2）、低（3）三个等级来表示，形成一个二维风险矩阵，而风险优先级可用二者值之和表示。这样，可能存在五个风险等级（即6、5、4、3、2，如图4-4所示）。

总之，风险优先级是由软件潜在问题影响的严重性确定的，是个相对值，而潜在问题的影响严重性是根据问题的可能性来评定的。

如图4-4中的风险优先级的确定是使用可能性和严重性等级值相加，但是如果使用两者值相乘，将会扩大有风险的区域。

综上所述，软件风险分析的目的是：确定测试对象、确定优先级，以及测试深度。在测试计划阶段，可以用风险分析的结果来确定软件测试的优先级。对每个测试项和测

试用例赋予优先级代码，将测试分为高、中和低的优先级类型，这样可以在有限的资源和时间条件下，合理安排测试的覆盖度与深度。

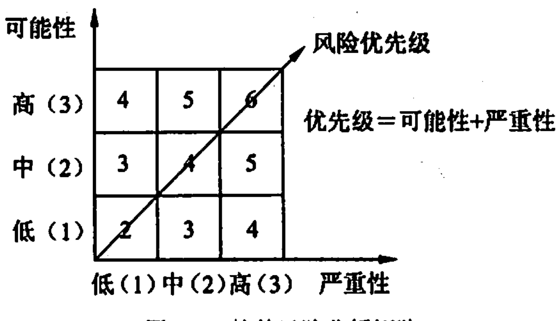


图 4-4 软件风险分析矩阵

4.8.4 软件测试风险

软件测试的风险是指软件测试过程出现的或潜在的问题，造成的原因主要是测试计划的不充分、测试方法有误或测试过程的偏离，造成测试的补充以及结果不准确。测试的不成功导致软件交付潜藏着问题，一旦在运行时爆发，会带来很大的商业风险。

美国 IEEE 829-1998《软件测试文档编制》标准中，在测试计划的模板中有一项为“风险与应急措施”。这表明软件测试风险管理是很重要的工作。主要是对测试计划执行的风险分析与制定要采取应急措施，降低软件测试产生的风险造成的危害。

测试计划的风险一般指测试进度滞后或出现非计划事件，就是针对计划好的测试工作造成消极影响的所有因素，对于计划风险分析的工作是制定计划风险发生时应采取的应急措施。一些常见的计划风险包括：交付日期、测试需求、测试范围、测试资源、人员的能力、测试预算、测试环境、测试支持、劣质组件、测试工具。

其中，交付日期的风险是主要风险之一。测试未按计划完成，发布日期推迟，影响对客户提交产品的承诺，管理的可信度和公司的信誉都要受到考验，同时也受到竞争对手的威胁。交付日期的滞后，也可能是已经耗尽了所有的资源。计划风险分析所做的工作重点不在于分析风险产生的原因，重点应放在提前制定应急措施来应对风险发生。当测试计划风险发生时，可能采用的应急措施有：缩小范围、增加资源、减少过程等措施。

比如，用户在软件开发接近尾声时，提出重要需求变动。

- 应急措施 1：增加资源。请求用户团队为测试工作提供更多的用户支持。
- 应急措施 2：缩小范围。决定在进行后续发布中实现较低优先级的特性。
- 应急措施 3：减少质量过程。在风险分析过程中确定某些风险级别低的特征测试或少测试。



上述列举的应急措施要涉及到有关方面的妥协：如果没有测试计划风险分析和应急措施处理风险，开发者和测试人员采取的措施比较匆忙，将不利于将风险的损失控制到最小。因此，软件风险分析和测试计划风险分析与应急措施是相辅相成的。综上所述，计划风险、软件风险、重点测试、不测试，甚至整个软件的测试与应急措施都是围绕“用风险来确定测试工作优先级”这样的原则来构造的。

4.9 软件测试的成本管理

有效的测试方法可以识别和评价软件的各种风险，能把这些风险缩小到测试范围内，我们可以承受的风险，并制定测试计划实现这个目标。

4.9.1 测试费用有效性

风险承受的确定，从经济学的角度考虑就是确定需要完成多少测试以及进行什么类型的测试。经济学所做的判断确定了软件存在的缺陷是否可以接受，如果可以，能承受多少。测试的策略不再主要由软件开发人员和测试人员来确定，而是由商业的经济利益来决定的。

“太少的测试是犯罪，而太多的测试是浪费。”对风险测试得过少，会造成软件的缺陷和系统的瘫痪；而对风险测试得过多，就会使本来没有缺陷的系统进行没有必要的测试，或者是对轻微缺陷的系统所花费的测试费用远远大于它们给系统造成的损失。

测试费用的有效性，可以用测试费用—质量曲线（如图 4-5所示）来表示。随着测试费用的增加，发现的缺陷也会越多，两线相交的地方是过多测试开始的地方，这时，排除缺陷的测试费用超过了缺陷给系统造成的损失费用。

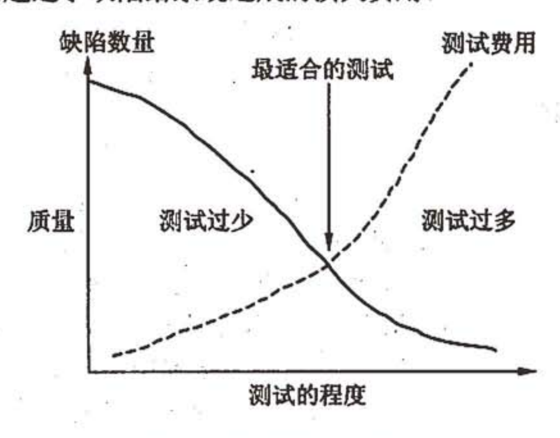


图 4-5 测试费用质量曲线

4.9.2 测试成本控制

在实际的软件测试中，资源条件是有限的，想要完成所有测试是不可能的。要么缺时间，要么缺钱和人，往往不知道实际测试成本有多少，也不知该怎样系统地降低成本。

测试工作的主要目标是使测试产能最大化，也就是，要使通过测试找出错误的能力最大化，而检测次数最小化。测试的成本控制目标是使测试开发成本、测试实施成本和测试维护成本最小化。

在软件产品开发过程中，各个阶段实施的测试成本并不很重要，有时可以看作是开发成本的一部分。但是，作为产品发布每一新版本而进行的重复性的测试所需的成本是主要考虑的问题。测试实施成本组成部分包括：测试准备成本、测试执行成本和测试结束成本。

1．测试准备成本控制

测试准备成本控制的目标是使时间消耗总量、劳动力总量，尤其是准备工作所需的熟练劳动力总量最小化。准备工作一般包括：硬件配置、软件配置、测试环境建立，以及测试环境的确定等。

2．测试执行成本控制

测试执行成本控制的目标是使总执行时间和所需的测试专用设备尽可能地减少。执行时间要求操作和用户进行手工操作执行测试时间应尽量减少，同时对劳动力和所需的技能也要尽量减少。如果需要重新测试，不同的选择会有不同的成本控制效果，重新测试的决策是在成本与风险的矛盾中进行的。

完全重新测试：将测试全部重新执行一遍，将风险降至最低，但加大了测试执行的成本。部分重新测试：有选择地重新执行部分测试，能减少执行成本，但同时加大了风险。

对部分重新测试进行合理的选择，将风险降至最低，而成本同样会很高，必须将其与测试执行成本进行比较，权衡利弊。利用测试自动化，进行重新测试，其成本效益是较好的。

部分重新测试选择方法有两种：

① 对由于程序变化而受到影响的每一部分进行重新测试；

② 对与变化有密切和直接关系的部分进行重新测试。

其中，第一种办法风险要小一些，而第二种是一种主观制定的办法，是建立在对软件产品十分了解的基础上的。一般地，选择重新测试的策略建立在软件测试错误的多少（即软件风险的大小）与测试的时间、人力、资源投入成本的大小之间的折衷基础上。



3．测试结束成本控制

测试结束成本的控制是进行测试结果分析和测试报告编制、测试环境的清除与恢复原环境所需的成本，使所需的时间和熟练劳动力总量减少到最低限度。

4．降低测试实施成本

测试准备环境的配置是十分重要的，要求与软件的运行环境相一致。测试环境应建立在固定的测试专用硬软件及网络环境中，尽可能使用软件和测试环境配置自动化。

测试实施尽可能采用自动化的测试工具，减少手工辅助测试。若测试执行需要人工，最好是请初级技术人员，而不是测试工程师。测试工程师一般是作为测试项目经理。

测试结束编制测试报告时，测试结果与预期结果的比较采用自动化方法，以降低分析比较成本。

测试自动化的方法主要有：使用测试工具；测试用例的自动化执行；测试文档编制的模板自动化生成。

5．降低测试维护成本

降低测试维护成本，与软件开发过程一样，加强软件测试的配置管理，所有测试的软件样品、测试文档（测试计划、测试说明、测试用例、测试记录、测试报告）都应置于配置管理系统控制之下。降低测试维护工作成本主要考虑：

- 对于测试中出现的偏差要增加测试；
- 采用渐进式测试以适应新变化的测试；
- 定期检查维护所有测试用例，以获得测试效果的连续性。

保持测试用例效果的连续性是重要的措施，有以下几个方面：

- 每一个测试用例都是可执行的，即被测产品，功能上不应有任何变化；
- 基于需求和功能的测试都应是适合的，若产品需求和功能发生小的变化，不应使测试用例无效；
- 每一个测试用例不断增加使用价值，即每一个测试用例不应是完全冗余的，连续使用应是成本效益高的。

4.9.3 质量成本

企业为了获得利润，需花费大量的资金进行测试。在质量方面的投资会产生利润，例如，提高产品质量会提高公司的声誉，使产品交付之后的维护成本减少，避免用户的报怨。测试是一种带有风险性的管理活动，减少企业在未来因为产品质量低劣而花费不必要的成本。

1．质量成本要素

（1）一致性成本（Cost of Conformance）

一致性成本是指用于保证软件质量的支出，包括预防成本（prevention cost）和测试预算，如测试计划、测试开发、测试实施费用。测试预算被称为审查费（appraisal cost）：

$$C_{Conformance} = C_{Prevention} + C_{Appraisal}$$

（2）非一致性成本（Cost of Nonconformance）

非一致成本是由出现的软件错误和测试过程故障（如延期、劣质的测试发布）引起的。这些问题会导致测试返工、补测、延迟。追加测试时间和资金就是一种由于内部故障引起的非一致成本。非一致成本还包括外部故障（软件遗留错误影响客户）引起部分。这些成本还包括技术支持小组预算，错误修正花费、产品收回、赔偿和销售成本。

$$C_{Nonconformance} = C_{Inter\text{-}failure} + C_{Exter\text{-}failure}$$

一般情况下，外部故障非一致成本要大于一致性成本与内部故障非一致成本之和。即：

$$C_{Exter\text{-}failure} > C_{Prevention} + C_{Appraisal} + C_{Inter\text{-}failure}$$

2．质量成本计算

质量成本= 一致性成本 + 非一致性成本，即：

$$C_{quality} = C_{Conformance} + C_{Nonconformance}$$

4.9.4　缺陷探测率（DDP Defect Detection Percentage）

缺陷探测率 DDP 是另一个衡量测试工作效率的软件质量成本的指标。

$$DDP = \frac{Bugs_{tester}}{Bugs_{tester} + Bugs_{customer}}$$

其中，$Bugs_{tester}$ 为测试者发现的错误数；$Bugs_{customer}$ 为客户发现并反馈技术支持人员进行修复的错误数。

缺陷探测率越高，也就是测试者发现的错误多，发布后客户发现的错误就越少，降低了外部故障不一致成本，达到了节约总成本的目的，可获得较高的测试投资回报率（ROI）。因此，缺陷探测率是衡量测试投资回报的一个重要指标。

4.9.5　测试投资回报举例

下面，通过一个案例来说明质量成本的概念。假设对一个开发的客户管理软件 CRM 进行测试。属于质量预防方面的一致性成本只考虑软件测试的投资，把发布之前和之后发现及修改的错误看成非一致性成本，假设发现的错误为 300 个，故障成本已知，测试过程的估算如下。

各阶段花费在发现及修改错误的成本假设如下：

① 在开发过程单元测试阶段，软件开发人员发现及修改一个错误需要 50 元；



② 建立独立的测试进行集成和系统测试，测试人员发现错误，开发人员修改后，测试人员再确认，一个错误需要300元；

③ 在产品发布后，由客户发现，报告技术支持人员、相关开发人员修改，测试组再进行回归测试，一个错误需要2000元。

第1种情况，开发单位未建立独立测试队伍，由开发人员进行测试，发现100个错误，而产品发布后客户发现错误200个，只存在故障成本构成的总成本为405000元，缺陷探测率为33.30%。

第2种情况，开发单位建立了独立测试队伍，进行手工测试。投资预算人员费用为60000元，测试环境使用费为8000元，测试投资（一致性成本）为68000元；除了开发过程中开发人员发现并修改100个错误外，测试过程中测试人员发现错误150个，而产品发布后客户发现50个错误。总质量成本下降到218000元（如表4-2所示），手工测试总质量成本节约了187000元，即为利润。投资回报率（ROI）为275%，缺陷探测率为83.3%。

$$
\begin{aligned}
\text{ROI} &= \frac{\text{节约的成本i} - \text{利润j}}{\text{测试投资}} \times 100\% \\
&= \frac{405000 - 218000}{68000} \times 100\% \\
&= 275\,\%
\end{aligned}
$$

$$
\begin{aligned}
\text{DDP} &= \frac{\text{Bugs}_{tester}}{\text{Bugs}_{tester} + \text{Bugs}_{customer}} \times 100\% \\
&= \frac{100 + 190}{100 + 190 + 10} \times 100\% \\
&= 83.3\,\%
\end{aligned}
$$

第3种情况，开发单位在独立测试中，采用自动测试工具，投资中增加10000元的工具使用费，测试投资为（一致性成本）78000元。由于使用测试工具，测试人员在测试中发现错误增加到190个，在产品发布后，客户发现错误下降到10个。总质量成本下降到160000元，比未建立独立测试前节约了245000元。投资回报率为314%，缺陷探测率为96.7%。

$$
\begin{aligned}
\text{ROI} &= \frac{405000 - 160000}{78000} \times 100\% \\
&= 314\%
\end{aligned}
$$

$$
\begin{aligned}
\text{DDP} &= \frac{100 + 190}{100 + 190 + 10} \times 100\% \\
&= 96.7\,\%
\end{aligned}
$$

综上所述，建立独立的软件测试，选择好的测试方案，不但软件缺陷的探测率高，有效地控制软件的风险，提高软件质量，而且降低了软件的质量成本，测试的投资回报率也将随着明显提高。

表 4-2　测试投资回报分析

质量成本项		测试成本项	开发测试	手工测试	自动测试
一致成本	测试投资	测试人工费		60000	60000
		环境使用费		8000	8000
		测试工具费			10000
		测试总投资		68000	78000
非一致成本	开发测试	发现错误数	100	100	100
		每个错误成本	100	100	100
		内部（开发）故障成本	5000	5000	5000
	独立测试	发现错误数		150	190
		每个错误成本		300	300
		内部（测试）故障成本		45000	57000
	客户支持	发现错误数	200	50	10
		每个错误成本	2000	2000	2000
		外部故障成本	400000	100000	20000
质量成本		一致性成本		68000	78000
		非一致性成本	405000	150000	82000
		总质量成本	405000	218000	160000
ROI		投资回报率	N/A	275%	314%
DDP		缺陷探测率	33.3%	83.3%	96.7%

第二篇　测 试 技 术

第 5 章　黑盒测试案例设计技术

5.1　概述

本章介绍黑盒测试的概念和进行黑盒测试的目的与意义，及关于等价类划分、边界值分析、因果图法、判定表法、正交试验法、功能图法等测试用例设计方法的原理与实现，并从测试设计说明、测试用例说明、测试程序说明三个方面介绍如何编写测试用例，最后结合一个 ATM 的例子体现如何设计测试用例。

5.2　测试用例设计方法

初涉软件测试者可能认为拿到软件后就可以立即进行测试，并希望马上找出软件的所有缺陷，这种想法就如同没有受过工程训练的开发工程师急于去编写代码一样。软件测试也是一个工程，也需要按照工程的角度去认识软件测试，在具体的测试实施之前，我们需要明白我们测试什么，怎么测试等，也就是说通过制定测试用例指导测试的实施。

5.2.1　什么是测试用例

所谓的测试用例设计就是将软件测试的行为活动，作一个科学化的组织归纳。软件测试是有组织性、步骤性和计划性的，而设计软件测试用例的目的，就是为了能将软件测试的行为转换为可管理的模式。软件测试是软件质量管理中最实际的行动，同时也是耗时最多的一项。基于时间因素的考虑，软件测试行为必须能够加以量化，才能进一步让管理阶层掌握所需要的测试过程，而测试用例就是将测试行为具体量化的方法之一。

简单地说，测试用例就是设计一个情况，软件程序在这种情况下，必须能够正常运行并且达到程序所设计的执行结果。如果程序在这种情况下不能正常运行，而且这种问题会重复发生，那就表示软件程序人员已经测出软件有缺陷，这时候就必须将这个问题标示出来，并且输入到问题跟踪系统内，通知软件开发人员。软件开发人员接获通知后，



将这个问题修改完成于下一个测试版本内，软件测试工程师取得新的测试版本后，必须利用同一个用例来测试这个问题，确保该问题已修改完成。

因为我们不可能进行穷举测试，为了节省时间和资源、提高测试效率，必须要从数量极大的可用测试数据中精心挑选出具有代表性或特殊性的测试数据来进行测试。

使用测试用例的好处主要体现在以下几个方面。

① 在开始实施测试之前设计好测试用例，可以避免盲目测试并提高测试效率。

② 测试用例的使用令软件测试的实施重点突出、目的明确。

③ 在软件版本更新后只需修正少部分的测试用例便可展开测试工作，降低工作强度，缩短项目周期。

④ 功能模块的通用化和复用化使软件易于开发，而测试用例的通用化和复用化则会使软件测试易于开展，并随着测试用例的不断精化其效率也不断攀升。

具体的黑盒测试用例设计方法包括等价类划分法、边界值分析法、错误推测法、因果图法、判定表驱动法、正交试验设计法、功能图法等。应该说，这些方法是比较实用的，但采用什么方法，在使用时自然要针对开发项目的特点对方法加以适当的选择。下面我们讨论几种常用的方法。

5.2.2 等价类划分法

等价类划分是一种典型的黑盒测试方法，用这一方法设计测试用例完全不考虑程序的内部结构，只根据对程序的要求和说明，即需求规格说明书。我们必须仔细分析和推敲说明书的各项需求，特别是功能需求。把说明中对输入的要求和输出的要求区别开来并加以分解。

由于穷举测试工作量太大，以至于无法实际完成，促使我们在大量的可能数据中选取其中的一部分作为测试用例。例如，在不了解等价分配技术的前提下，我们做计算器程序的加法测试时，测试了 1+1，1+2，1+3 和 1+4 之后，还有必要测试 1+5 和 1+6 吗，能否放心地认为它们是正确的？我们感觉 1+5 和 1+6，与前面的 1+1，1+2 都是很类似的简单加法。

等价类划分的办法是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则这一类中的其他例子也不会查出错误（除非等价类中的某些例子属于另一等价类，因为几个等价类是可能相交的）。使用这一方法设计测试用例，首先必须在分析需求规格说明的基础上划分等价类，列出等价类表。

1．划分等价类和列出等价类表

等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的。并合理地假定：测试某等价类的代表值就等于对这一类其他值的测试。

因此，可以把全部输入数据合理地划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件，就可以用少量代表性的测试数据取得较好的测试结果。等价类划分有两种不同的情况：有效等价类和无效等价类。

有效等价类：指对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。

无效等价类：与有效等价类的定义恰巧相反。

设计测试用例时，要同时考虑这两种等价类。因为软件不仅要能接收合理的数据，也要能经受意外的考验。这样的测试才能确保软件具有更高的可靠性。

下面给出 6 条确定等价类的原则：

① 在输入条件规定了取值范围或值的个数的情况下，可以确立一个有效等价类和两个无效等价类。

② 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可以确立一个有效等价类和一个无效等价类。

③ 在输入条件是一个布尔量的情况下，可确定一个有效等价类和一个无效等价类。

④ 在规定了输入数据的一组值（假定 n 个），并且程序要对每一个输入值分别处理的情况下，可确立 n 个有效等价类和一个无效等价类。

⑤ 在规定了输入数据必须遵守的规则的情况下，可确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

⑥ 在确知已划分的等价类中，各元素在程序处理中的方式不同的情况下，则应再将该等价类进一步地划分为更小的等价类。

在确立了等价类之后，建立等价类表，列出所有划分出的等价类如表 5-1 所示。

表 5-1　等价类表示例

输入条件	有效等价类	无效等价类	输入条件	有效等价类	无效等价类
…	…	…	…	…	…

2．确定测试用例

根据已列出的等价类表，按以下步骤确定测试用例：

① 为每个等价类规定一个惟一的编号。

② 设计一个新的测试用例，使其尽可能多地覆盖尚未覆盖的有效等价类。重复这

一步，最后使得所有有效等价类均被测试用例所覆盖。

③ 设计一个新的测试用例，使其只覆盖一个无效等价类。重复这一步使所有无效等价类均被覆盖。

在寻找等价区间时，想办法把软件的相似输入、输出、操作分成组。这些组就是等价区间。请看一些例子。

在两数相加用例中，测试 1+13 和 1+99999999 似乎有点不同。这是一种直觉，一个是普通加法，而另一个似乎有些特殊，这个直觉是对的。程序对 1 和最大数值相加的处理和对两个小一些的数值相加的处理有所不同。后者必须处理溢出情况。因为软件操作可能不同，所以这两个用例属于不同的等价区间。

如果具有编程经验，就可能会想到更多可能导致软件操作不同的“特殊”数值。如果不是程序员，也不用担心，你很快就会学到这种技术，无须了解代码细节就可以运用。

如图 5-1 所示是复制的多种方法，给出了选中编辑菜单后显示复制和粘贴命令的计算器程序。每一项功能（即复制和粘贴）有 5 种执行方式。要想复制，可以单击复制菜单命令，键入 C，按 Ctrl+C 或 Ctrl+Shift+C 组合键。任何一种输入途径都会把当前数值复制到剪贴板中，一一执行同样的输出操作，产生同样的结果。

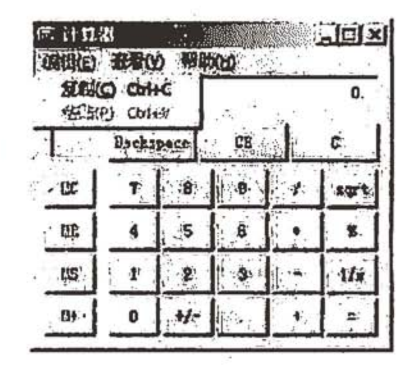


图 5-1 复制的多种方法

如果要测试复制命令，可以把这 5 种输入途径划分减为 3 个，单击菜单命令，键入 C 和按 Ctrl+C 组合键。对软件质量有了信心之后，知道无论以何种方式激活复制功能都工作正常，甚至可以进一步缩减为 1 个区间，例如按 Ctrl+C 组合键。

再看下一个例子。看一下在标准的另存为对话框（如图 5-2 所示）中输入文件名称的情形。

Windows 文件名可以包含除了“、”、“/”、“：”、“ •”、“?”、“<>”和“\”之外的任意字符。文件名长度是 1～255 个字符。如果为文件名创建测试用例，等价区间有合法字符、非法字符、合法长度的名称、过长名称和过短名称。

例题：根据下面给出的规格说明，利用等价类划分的方法，给出足够的测试用例。“一个程序读入 3 个整数，把这 3 个数值看作一个三角形的 3 条边的长度值。这个程序要打印出信息，说明这个三角形是不等边的、是等腰的、还是等边的”。

我们可以设三角形的 3 条边分别为 A，B，C。如果它们能够构成三角形的 3 条边，必须满足：

A>0，B>0，C>0，且 A+B>C，B+C>A，A+C>B。

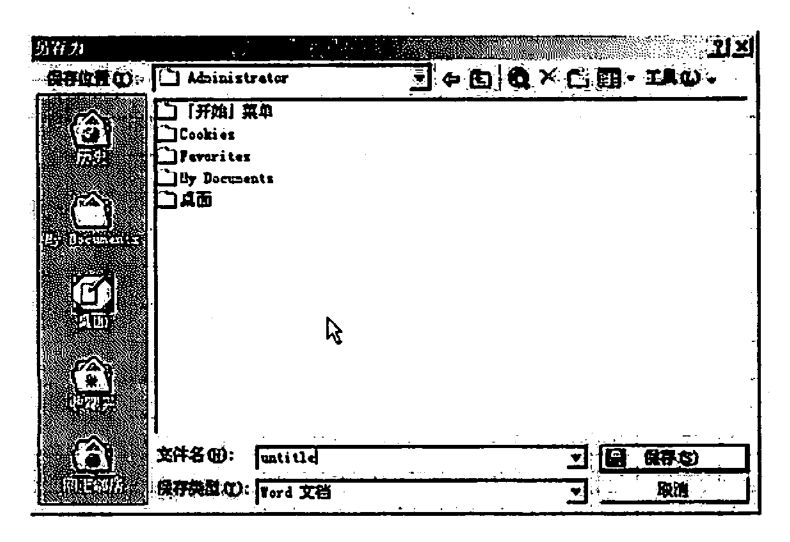


图 5-2　存盘对话框

如果是等腰的，还要判断 A=B，或 B=C，或 A=C。

如果是等边的，则需判断是否 A=B，且 B=C，且 A=C。

列出等价类表，如表 5-2 所示。

表 5-2　等 价 类 表

输 入 条 件	有效等价类	无效等价类
是否三角形的 3 条边	(A>0)，　(1) (B>0)，　(2) (C>0)，　(3) (A+B>C)，　(4) (B+C>A)，　(5) (A+C>B)　(6)	(A≤0)，　(7) (B≤0)，　(8) (C≤0)，　(9) (A+B≤C)，　(10) (B+C≤A)，　(11) (A+C≤B)　(12)
是否等腰三角形	(A=B)，　(13) (B=C)，　(14) (C=A)，　(15)	(A≠B) and (B≠C) and (C≠A)， (16)
是否等边三角形	(A=B) and (B=C) and (C=A)， (17)	(A≠B)，　(18) (B≠C)，　(19) (C≠A)，　(20)

设计测试用例：输入顺序是【A，B，C】，如表 5-3 所示。



表 5-3 测试用例

<table>
<tr><th>序号</th><th>【A，B，C】</th><th>覆盖等价类</th><th>输　出</th></tr>
<tr><td>1</td><td>【3，4，5】</td><td>（1），（2），（3），（4），（5），（6）</td><td>一般三角形</td></tr>
<tr><td>2</td><td>【0，1，2】</td><td>（7）</td><td rowspan="6">不能构成三角形</td></tr>
<tr><td>3</td><td>【1，0，2】</td><td>（8）</td></tr>
<tr><td>4</td><td>【1，2，0】</td><td>（9）</td></tr>
<tr><td>5</td><td>【1，2，3】</td><td>（10）</td></tr>
<tr><td>6</td><td>【1，3，2】</td><td>（11）</td></tr>
<tr><td>7</td><td>【3，1，2】</td><td>（12）</td></tr>
<tr><td>8</td><td>【3，3，4】</td><td>（1），（2），（3），（4），（5），（6），（13）</td><td rowspan="3">等腰三角形</td></tr>
<tr><td>9</td><td>【3，4，4】</td><td>（1），（2），（3），（4），（5），（6），（14）</td></tr>
<tr><td>10</td><td>【3，4，3】</td><td>（1），（2），（3），（4），（5），（6），（15）</td></tr>
<tr><td>11</td><td>【3，4，5】</td><td>（1），（2），（3），（4），（5），（6），（16）</td><td>非等腰三角形</td></tr>
<tr><td>12</td><td>【3，3，3】</td><td>（1），（2），（3），（4），（5），（6），（17）</td><td>是等边三角形</td></tr>
<tr><td>13</td><td>【3，4，4】</td><td>（1），（2），（3），（4），（5），（6），（14），（18）</td><td rowspan="3">非等边三角形</td></tr>
<tr><td>14</td><td>【3，4，3】</td><td>（1），（2），（3），（4），（5），（6），（15），（19）</td></tr>
<tr><td>15</td><td>【3，3，4】</td><td>（1），（2），（3），（4），（5），（6），（13），（20）</td></tr>
</table>

请记住，等价分配的目标是把可能的测试用例组合缩减到仍然足以满足软件测试需求为止。因为，选择了不完全测试，就要冒一定的风险，所以必须仔细选择分类。

关于等价分配最后要讲的一点是，这样做有可能不客观。科学有时也是一门艺术。测试同一个复杂程序的两个软件测试员，可能会制定出两组不同的等价区间。只要审查等价区间的人都认为它们足以覆盖测试对象就可以了。

5.2.3 边界值分析法

人们从长期的测试工作经验得知，大量的错误是发生在输入或输出范围的边界上的，而不是在输入范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。例如，在做三角形计算时，要输入三角形的 3 个边长 A、B 和 C。这 3 个数值应当满足 A>0、B>0、C>0、A+B>C、A+C>B、B+C>A，才能构成三角形。但如果把 6 个不等式中的任何一个大于号“>”错写成大于等于号“≥”，那就不能构成三角形。问题恰恰出现在容易被疏忽的边界附近。这里所说的边界是指相当于输入等价类和输出等价类而言，稍高于其边界值及稍低于其边界值的一些特定情况。

1．边界条件

我们可以想象一下，如果在悬崖峭壁边可以自信地安全行走，平地就不在话下了。如果软件在能力达到极限时能够运行，那么在正常情况下一般也就不会有什么问题。

边界条件是特殊情况，因为编程从根本上说不怀疑边界有问题。奇怪的是，程序在处理大量中间数值时都是对的，但是可能在边界处出现错误。下面的一段源代码说明了在一个极简单的程序中是如何产生边界条件问题的。

```
rem create a 10 element integer array
rem initialize each element to–1
dim data（10）as integer
dim i as integer
for i=1 to 10
data（i）=–1
next i
end
```

这段代码的意图是创建包含 10 个元素的数组，并为数组中的每一个元素赋初值–1。看起来相当简单。它建立了包含 10 个整数的数组 data 和一个计数值 i。For 循环是从 1～10，数组中从第 1 个元素到第 10 个元素被赋予数值–1。那么边界问题在哪儿呢？

在大多数开发语言脚本中，应当以声明的范围定义数组，在本例中定义语句是 dim data（10）as interger，第一个创建的元素是 data（0），而不是 data（1）。该程序实际上创建了一个从 data（0）～ data（10）共 11 个元素的数组。程序从 1～10 循环将数组元素的值初始化为-1，但是由于数组的第一个元素是 data（0），因此它没有被初始化。程序执行完毕，数组值如下：

```
data(0)= 0      data(6)= –1
data(1)= –1     data(7)= –1
data(2)= –1     data(8)= –1
data(3)= –1     data(9)= –1
data(4)= –1     data(10)= –1
data(5)= –1
```

注意 data（0）的值是 0，而不是–1。如果这位程序员以后忘记了，或者其他程序员不知道这个数据数组是如何初始化的，那么他就可能会用到数组的第 1 个元素 data（0），以为它的值是–1。诸如此类的问题很常见，在复杂的大型软件中，可能导致极其严重的软件缺陷。

2．次边界条件

上面讨论的普通边界条件是最容易找到的。它们在产品说明书中有定义，或者在使用软件的过程中确定。而有些边界在软件内部，最终用户几乎看不到，但是软件测试仍有必要检查。这样的边界条件称为次边界条件或者内部边界条件。



寻找这样的边界不要求软件测试员具有程序员那样阅读源代码的能力，但是要求大体了解软件的工作方式。2 的乘方和 ASCII 表就是这样的例子。

（1）2 的乘方

计算机和软件的计数基础是二进制数，用位（bit）来表示 0 和 1，一个字节（byte）由 8 位组成，一个字（word）由两个字节组成等。表 5-4 中列出了常用的 2 的乘方单位及其范围或值。

表 5-4 软件中 2 的乘方

术　语	范围或值	术　语	范围或值
位	0 或 1	千	1,024
双位	0～15	兆	1,048,576
字节	0～255	亿	1,073,741,824
字	0～65,535	万亿	1,099,511,627,776

表 5-4 中所列的范围和值是作为边界条件的重要数据。除非软件向用户提出这些范围，否则在需求文档中不会指明。然而，它们通常由软件内部使用，外部是看不见的，当然，在产生软件缺陷的情况下可能会看到。

在建立等价区间时，要考虑是否需要包含 2 的乘方边界条件。例如，如果软件接受用户输入 1～1000 范围内的数字，谁都知道在合法区间中包含 1 和 1000，也许还要有 2 和 999。为了覆盖任何可能的 2 的乘方次边界，还要包含临近双位边界的 14、15 和 16，以及临近字节边界的 254、255 和 256。

（2）ASCII 表

另一个常见的次边界条件是 ASCII 字符表。如表 5-5 所示是部分 ASCII 值表的清单。

表 5-5 部分 ASCII 值表

字符	ASCII 值	字符	ASCII 值	字符	ASCII 值	字符	ASCII 值
Null	0	B	66	2	50	a	97
Space	32	Y	89	9	57	b	98
/	47	Z	90	:	58	y	121
0	48	[	91	@	64	z	122
1	49	'	96	A	65	{	123

注意，表 5-5 不是结构良好的连续表。0～9 的后面 ASCII 值是 48～57。斜杠字符（/）在数字 0 的前面，而冒号字符“:”在数字 9 的后面。大写字母 A～Z 对应 65～90。小写字母对应 97～122。这些情况都代表次边界条件。

如果测试进行文本输入或文本转换的软件，在定义数据区间包含哪些值时，参考一

下 ASCII 表是相当明智的。例如，如果测试的文本框只接受用户输入字符 A～Z 和 a～z，就应该在非法区间中包含 ASCII 表中这些字符前后的值@、[、和{ 。

（3）其他一些边界条件

另一种看起来很明显的软件缺陷来源是当软件要求输入时（比如在文本框中），不是没有输入正确的信息，而是根本没有输入任何内容，只按了 Enter 键。这种情况在产品说明书中常常被忽视，程序员也可能经常遗忘，但是在实际使用中却时有发生。程序员总会习惯性地认为用户要么输入信息，不管是看起来合法的或非法的信息，要么就会选择 Cancel 键放弃输入，如果没有对空值进行好的处理的话，恐怕程序员自己都不知道程序会引向何方。

正确的软件通常应该将输入内容默认为合法边界内的最小值，或者合法区间内的某个合理值，否则，返回错误提示信息。

因为这些值通常在软件中进行特殊处理，所以不要把它们与合法情况和非法情况混在一起，而要建立单独的等价区间。

3．边界值的选择方法

边界值分析是一种补充等价划分的测试用例设计技术，它不是选择等价类的任意元素，而是选择等价类边界的测试用例。实践证明，为检验边界附近的处理专门设计测试用例，常常取得良好的测试效果。边界值分析法不仅重视输入条件边界，而且也适用于输出域测试用例。

对边界值设计测试用例，应遵循以下几条原则：

① 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。

② 如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少 1、比最大个数多 1 的数作为测试数据。

③ 根据规格说明的每个输出条件，使用前面的原则①。

④ 根据规格说明的每个输出条件，应用前面的原则②。

⑤ 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素作为测试用例。

⑥ 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构边界上的值作为测试用例。

⑦ 分析规格说明，找出其他可能的边界条件。

5.2.4　错误推测法

错误推测法就是基于经验和直觉推测程序中所有可能存在的各种错误，有针对性地

设计测试用例的方法。

错误推测法的基本思想是列举出程序中所有可能有的错误和容易发生错误的特殊情况，根据它们选择测试用例。例如，设计一些非法、错误、不正确和垃圾数据进行输入测试是很有意义的。如果软件要求输入数字，就输入字母。如果软件只接受正数，就输入负数。如果软件对时间敏感，就看它在公元 3000 年是否还能正常工作。还有，例如，在单元测试时曾列出的许多在模块中常见的错误，以前产品测试中曾经发现的错误等，这些就是经验的总结。另外，输入数据和输出数据为 0 的情况，或者输入表格为空格或输入表格只有一行，这些都是容易发生错误的情况。可选择这些情况下的例子作为测试用例。

5.2.5 因果图法

前节介绍的等价类划分方法和边界值分析法都是着重考虑输入条件，并没有考虑到输入情况的各种组合，也没考虑到各个输入情况之间的相互制约关系。如果在测试时必须考虑输入条件的各种组合，可能的组合数将是天文数字。因此必须考虑描述多种条件的组合，相应地产生多个动作的形式来考虑设计测试用例，这就需要利用因果图。在软件工程中，有些程序的功能可以用判定表的形式来表示，并根据输入条件的组合情况规定相应的操作。很自然，应该为判定表中的每一列设计一个测试用例，以便保证测试程序在输入条件的某种组合下，操作是正确的。

1. 因果图设计方法

因果图法是从用自然语言书写的程序规格说明的描述中找出因(输入条件)和果(输出或程序状态的改变)，通过因果图转换为判定表。

利用因果图导出测试用例需要经过以下几个步骤：

① 分析程序规格说明的描述中，哪些是原因，哪些是结果。原因常常是输入条件或是输入条件的等价类，而结果是输出条件。

② 分析程序规格说明的描述中语义的内容，并将其表示成连接各个原因与各个结果的“因果图”。

③ 标明约束条件。由于语法或环境的限制，有些原因和结果的组合情况是不可能出现的。为表明这些特定的情况，在因果图上使用若干个标准的符号标明约束条件。

④ 把因果图转换成判定表。

⑤ 为判定表中每一列表示的情况设计测试用例。

因果图生成的测试用例（局部，组合关系下的）包括了所有输入数据的取 TRUE 与取 FALSE 的情况，构成的测试用例数目达到最少，且测试用例数目随输入数据数目的增加而增加。

事实上，在较为复杂的问题中，这个方法常常是十分有效的，它能有力地帮助我们确定测试用例。当然，如果哪个开发项目在设计阶段就采用了判定表，也就不必再画因果图了，而是可以直接利用判定表设计测试用例了。

通常在因果图中，用 C_i 表示原因，E_i 表示结果，其基本符号如图 5-3 所示。各结点表示状态，可取“0”或“1”值。“0”表示某状态不出现，“1”表示某状态出现。

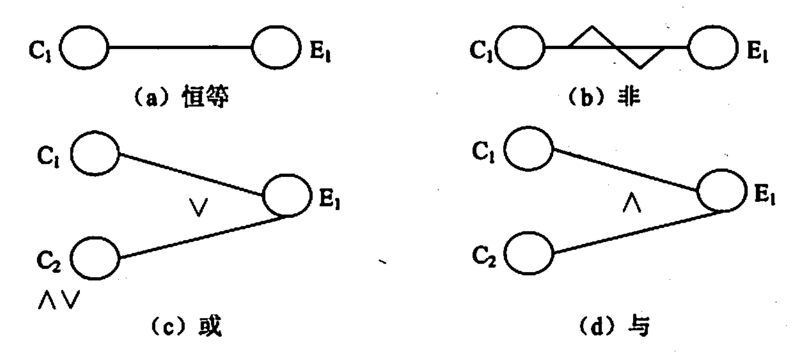


图 5-3　因果图的基本图形符号

① 恒等：若原因出现，则结果出现；若原因不出现，则结果也不出现。

② 非（～）：若原因出现，则结果不出现；若原因不出现，则结果出现。

③ 或（∨）：若几个原因中有 1 个出现，则结果出现；若几个原因都不出现，则结果不出现。

④ 与（∧）：若几个原因都出现，结果才出现。若其中有 1 个原因不出现，则结果不出现。

为了表示原因与原因之间、结果与结果之间可能存在的约束条件，在因果图中可以附加一些表示约束条件的符号。从输入（原因）考虑，有 4 种约束，例如：(a)、(b)、(c)、(d)。从输出（结果）考虑，还有 1 种约束，例如：(e)，如图 5-4 所示。

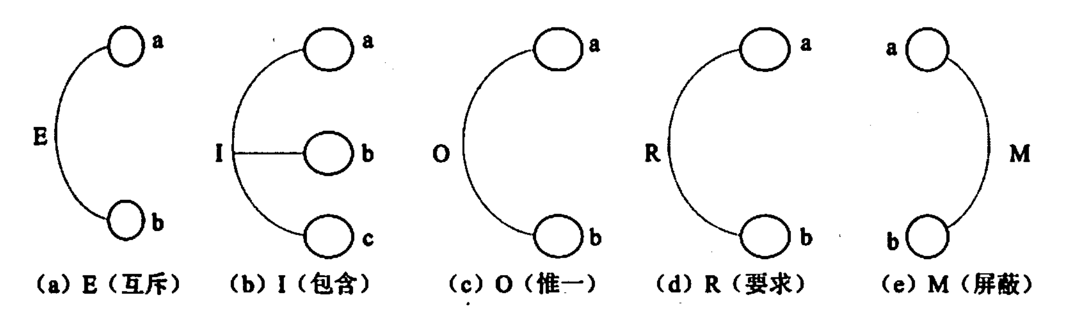


图 5-4　因果图的约束符号



① E（互斥）：表示 a、b 两个原因不会同时成立，两个中最多有一个可能成立。

② I（包含）：表示 a、b、c 这 3 个原因中至少有一个必须成立。

③ O（惟一）：表示 a 和 b 当中必须有一个，且仅有一个成立。

④ R（要求）：表示当 a 出现时，b 必须也出现。a 出现时不可能 b 不出现。

⑤ M（屏蔽）：表示当 a 是 1 时，b 必须是 0。而当 a 为 0 时，b 的值不定。

2．因果图测试用例

例如：有一个处理单价为 1 元 5 角钱的盒装饮料的自动售货机软件。若投入 1 元 5 角硬币，按下“可乐”、“雪碧”或“红茶”按钮，相应的饮料就送出来。若投入的是两元硬币，在送出饮料的同时退还 5 角硬币。

分析这一段说明，我们可以列出原因和结果。

原因：① 投入 1 元 5 角硬币；② 投入 2 元硬币；
③ 按“可乐”按钮；④ 按“雪碧”按钮；⑤ 按“红茶”按钮。

中间状态：① 已投币；② 已按钮。

结果：① 退还 5 角硬币；② 送出“可乐”饮料；
③ 送出“雪碧”饮料；④ 送出“红茶”饮料。

根据原因和结果，我们可以设计这样一个因果图（如图 5-5 所示。）

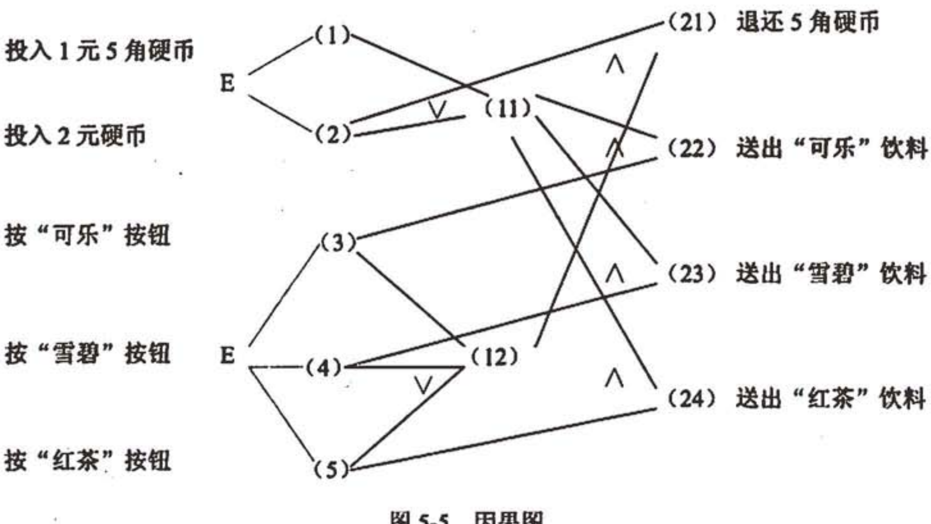


图 5-5 因果图

转换为测试用例，如表 5-6 所示，每一列可作为确定测试用例的依据。

表 5-6

			1	2	3	4	5	6	7	8	9	10	11
输入	投入 1 元 5 角硬币	(1)	1	1	1	1	0	0	0	0	0	0	0
	投入 2 元硬币	(2)	0	0	0	0	1	1	1	1	0	0	0
	按“可乐”按钮	(3)	1	0	0	0	1	0	0	0	1	0	0
	按“雪碧”按钮	(4)	0	1	0	0	0	1	0	0	0	1	0
	按“红茶”按钮	(5)	0	0	1	0	0	0	1	0	0	0	1
中间结点	已投币	(11)	1	1	1	1	1	1	1	1	0	0	0
	已按钮	(12)	1	1	1	0	1	1	1	0	1	1	1
输出	退还 5 角硬币	(21)	0	0	0	0	1	1	1	0	0	0	0
	送出“可乐”饮料	(22)	1	0	0	0	1	0	0	0	0	0	0
	送出“雪碧”饮料	(23)	0	1	0	0	0	1	0	0	0	0	0
	送出“红茶”饮料	(24)	0	0	1	0	0	0	1	0	0	0	0

5.2.6 判定表驱动法

前面因果图方法中已经用到了判定表。判定表是分析和表达多逻辑条件下执行不同操作的情况的工具。在程序设计发展的初期，判定表就已被用作编写程序的辅助工具了。它可以把复杂的逻辑关系和多种条件组合的情况表达得较明确。

1. 判定表组成

判定表通常由 4 个部分组成，如图 5-6 所示。

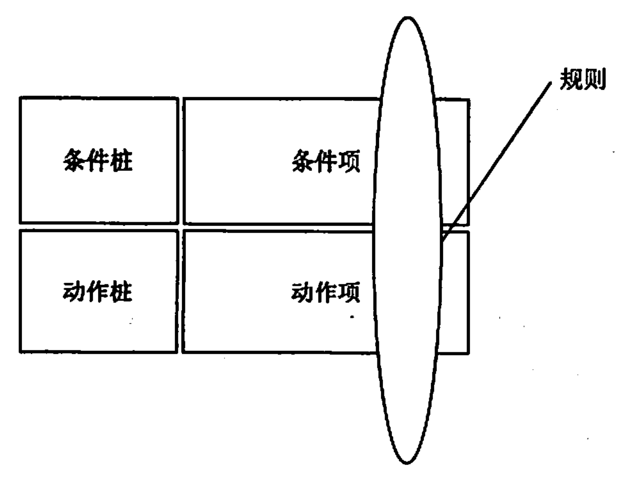


图 5-6　判定表



- 条件桩（condition stub）：列出了问题的所有条件。通常认为列出的条件的次序无关紧要。
- 动作桩（action stub）：列出了问题规定可能采取的操作。这些操作的排列顺序没有约束。
- 条件项（condition entry）：列出针对它所列条件的取值，在所有可能情况下的真假值。
- 动作项（action entry）：列出在条件项的各种取值情况下应该采取的动作。
- 规则：任何一个条件组合的特定取值及其相应要执行的操作。在判定表中贯穿条件项和动作项的一列就是一条规则。显然，判定表中列出多少组条件取值，也就有多少条规则，条件项和动作项就有多少列。

2．判定表建立

判定表的建立因该依据软件规格说明，步骤如下：

① 确定规则的个数。假如有 n 个条件，每个条件有两个取值（0，1），故有 2n 种规则。

② 列出所有的条件桩和动作桩。

③ 填入条件项。

④ 填入动作项。制定初始判定表。

⑤ 简化。合并相似规则或者相同动作。

Beizer 指出了适合使用判定表设计测试用例的条件：

① 规格说明以判定表的形式给出，或很容易转换成判定表。

② 条件的排列顺序不影响执行哪些操作。

③ 规则的排列顺序不影响执行哪些操作。

④ 当某一规则的条件已经满足，并确定要执行的操作后，不必检验别的规则。

⑤ 如果某一规则要执行多个操作，这些操作的执行顺序无关紧要。

5.2.7 正交试验法

利用因果图来设计测试用例时，作为输入条件的原因与输出结果之间的因果关系，有时很难从软件需求规格说明中得到。往往因果关系非常庞大，导致利用因果图而得到的测试用例数目多得惊人，给软件测试带来沉重的负担。为了有效地、合理地减少测试的工时与费用，可利用正交试验法进行测试用例的设计。

1．正交试验设计方法

依据 Galois 理论，正交试验设计方法是从大量的试验数据中挑选适量的、有代表性的点，从而合理地安排测试的一种科学的试验设计方法。

正交试验法，就是使用已经造好了的表格“——”正交表来安排试验并进行数据分析的一种方法。它简单易行并且计算表格化，应用性较好。下边通过一个例子来说明正交试验法。

例题：为提高某化工产品的转化率，选择了三个有关因素进行条件试验，反应温度（A），反应时间（B），用碱量（C），并确定了它们的试验范围如下。

- A：80～90℃；
- B：90～150 分钟；
- C：5%～7%。

试验目的是搞清楚因子 A、B、C 对转化率有什么影响，哪些是主要的，哪些是次要的，从而确定最适生产条件，即温度、时间及用碱量各为多少才能使转化率最高。这里，对因子 A、B 和 C，在试验范围内都选了三个水平，如下所示。

- A：A1=80℃，A2=85℃，A3=90℃；
- B：B1=90 分钟，B2=120 分钟，B3=150 分钟；
- C：C1=5%，C2=6%，C3=7%。

当然，在正交试验设计中，因子可以是定量的，也可以是定性的。而定量因子各水平间的距离可以相等，也可以不相等。这个三因子三水平的条件试验，通常有两种试验方法：

① 取三因子所有水平之间的组合，即 A1B1C1，A1B1C2，A1B2C1，……，A3B3C3，共有 33=27 次试验。用图 5-7 表示立方体的 27 个节点。这种试验法叫做全面试验法。

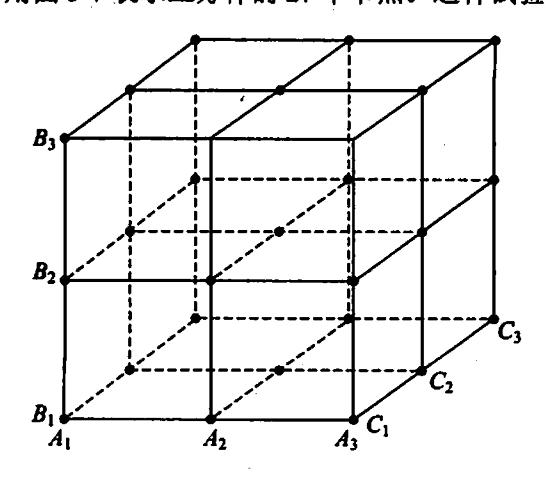


图 5-7 全面试验法取点

全面试验对各因子与指标间的关系剖析得比较清楚。但试验次数太多。特别是当因



子数目多，每个因子的水平数目也很多时，试验量非常大。如选 6 个因子，每个因子取 5 个水平时，如欲做全面试验，则需 5^6=15625 次试验，这实际上是不可能实现的。如果应用将要介绍的正交试验法，只做 25 次试验就行了。而且在某种意义上讲，这 25 次试验代表了 15625 次试验。

② 简单对比法，即变化一个因素而固定其他因素，如首先固定 B、C 于 Bl、Cl，使 A 变化：

↗A1
B1C1 →A2
↘A3（好结果）

如得出结果 A3 最好，则固定 A 于 A3，C 还是 Cl，使 B 变化：

↗B1
A3C1 →B2 （好结果）
↘B3

得出结果以 B2 为最好，则固定 B 于 B2，A 于 A3，使 C 变化：

↗C1
A3B2→C2 （好结果）
↘C3

试验结果以 C2 为最好。于是就认为最好的工艺条件是 A3B2C2。

这种方法一般也有一定的效果，但缺点很多。首先这种方法的选点代表性很差，如按上述方法进行试验，试验点完全分布在一个角上，而在一个很大的范围内没有选点，因此这种试验方法不全面，所选的工艺条件 A3B2C2 不一定是 27 个组合中最好的。其次，用这种方法比较条件好坏时，是把单个的试验数据拿来，进行数值上的简单比较，而试验数据中必然包含着误差成分，所以单个数据的简单比较不能剔除误差，必然造成结论的不稳定。

简单对比法的最大优点就是试验次数少，例如，6 因子 5 水平试验，在不重复时，只用 5+（6–1）×（5–1）=5+5×4=25 次试验就可以了。

考虑兼顾这两种试验方法的优点，从全面试验的点中选择具有典型性、代表性的点，使试验点在试验范围内分布得很均匀，能反映全面情况。但我们又希望试验点尽量地少，为此还要具体考虑一些问题。 如上例，对应于 A 有 Al、A2、A3 3 个平面，对应于 B、C 也各有 3 个平面，共 9 个平面。则这 9 个平面上的试验点都应当一样多，即对每个因子的每个水平都要同等看待。具体来说，每个平面上都有 3 行、3 列，要求在每行、每列上的点一样多。这样，作出如图 5-8 所示的设计，试验点用⊙表示。我

们看到，在 9 个平面中每个平面上都恰好有 3 个点，而每个平面的每行每列都有 1 个点，而且只有 1 个点，总共 9 个点。这样的试验方案，试验点的分布很均匀，试验次数也不多。

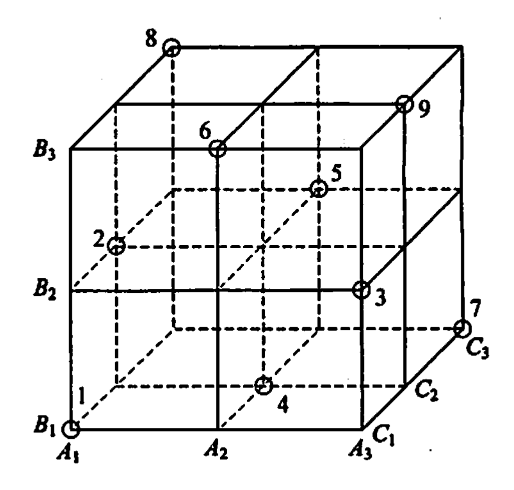


图 5-8　正交试验设计图例

当因子数和水平数都不太大时，尚可通过作图的办法来选择分布很均匀的试验点。但是因子数和水平数多了，作图的方法就不行了。试验工作者在长期的工作中总结出一套办法，创造出所谓的正交表。按照正交表来安排试验，既能使试验点分布得很均匀，又能减少试验次数，而且计算分析简单，能够清晰地阐明试验条件与指标之间的关系。用正交表来安排试验及分析试验结果，这种方法叫正交试验设计法。

一般用 L 代表正交表，常用的有 L_8（2^7）、L_9（3^4）、L_{16}（4^5）、L_8（$4×2^4$）等。此符号各数字的意义如下。

例如：L_8（2^7），其中，7 为此表列的数目（最多可安排的因子数）；2 为因子的水平数；8 为此表行的数目（试验次数）。

又例如：L_{18}（$2×3^7$），有 7 列是 3 水平的，有 1 列是 2 水平的，L_{18}（$2×3^7$）的数字告诉我们，用它来安排试验，做 18 个试验最多可以考察 1 个 2 水平因子和 7 个 3 水平因子。

在行数为 mn 型的正交表中（m，n 是正整数），试验次数（行数）＝∑（每列水平数−1）+1，如 L_8（2^7），8＝7×（2−1）+1，利用上述关系式可以从所要考察的因子水平数来决定最低的试验次数，进而选择合适的正交表。比如要考察 5 个 3 水平因子及一个 2 水平因子，则起码的试验次数为 5×（3−1）+1×（2−1）+1＝12（次），这就是说，要在行数不小于 12，既有 2 水平列又有 3 水平列的正交表中选择，L_{18}（$2×3^7$）适合。正交表具有两条性质：每一列中各数字出现的次数都一样多；任何两列所构成的各有序数对出现的次数都一样多。所以称之为正交表。

例如，在 L_9（3^4）中（如表 5-7 所示），各列中的 1、2、3 都各自出现 3 次；任何两列，例如第 3、4 列，所构成的有序数对从上向下共有 9 种，既没有重复也没有遗漏。其他任何两列所构成的有序数对也是这 9 种各出现一次。这反映了试验点分布的均匀性。



表 5-7 $L_9(3^4)$ 正交表

行号	列号 1	列号 2	列号 3	列号 4
	水平			
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

试验方案应该如何设计呢？安排试验时，只要把所考察的每一个因子任意地对应于正交表的一列（一个因子对应一列，不能让两个因子对应同一列），然后把每列的数字"翻译"成所对应因子的水平。这样，每一行的各水平组合就构成了一个试验条件（不考虑没安排因子的列）。对于上例，因子 A、B、C 都是 3 水平的，试验次数要不少于 3×（3−1）+1＝7（次），可考虑选用 $L_9(3^4)$。因子 A、B、C 可任意地对应于 $L_9(3^4)$ 的某三列，例如 A、B、C 分别放在 1、2、3 列，然后试验按行进行，顺序不限，每一行中各因素的水平组合就是每一次的试验条件，从上到下就是这个正交试验的方案，如表 5-8 所示。这个试验方案的几何解释正好是正交试验设计图例。

表 5-8 试验方案

行号 \ 列号	A 1	B 2	C 3	4	⇨	试验号	水平组合	试验条件 温度（℃）	时间（分）	加碱量（%）
1	1	1	1	1		1	$A_1B_1C_1$	80	90	5
2	1	2	2	2		2	$A_1B_2C_2$	80	120	6
3	1	3	3	3		3	$A_1B_3C_3$	80	150	7
4	2	1	2	3		4	$A_2B_1C_2$	85	90	6
5	2	2	3	1		5	$A_2B_2C_3$	85	120	7
6	2	3	1	2		6	$A_2B_3C_1$	85	150	5
7	3	1	3	2		7	$A_3B_1C_3$	90	90	7
8	3	2	1	3		8	$A_3B_2C_1$	90	120	5
9	3	3	2	1		9	$A_3B_3C_2$	90	150	6

3 个 3 水平的因子，做全面试验需要 $3^3=27$ 次试验，现用 $L_9(3^4)$ 来设计试验方案，只要做 9 次，工作量减少了 2/3，而在一定意义上代表了 27 次试验。

2．正交试验测试用例设计步骤

利用正交试验设计测试用例的步骤如下。

- 提取功能说明，构造因子“——”状态表。把影响实验指标的条件称为因子，而影响实验因子的条件叫做因子的状态。利用正交试验设计方法来设计测试用例时，首先要根据被测试软件的规格说明书找出影响其功能实现的操作对象和外部因素，把它们当作因子，而把各个因子的取值当做状态。对软件需求规格说明中的功能要求进行划分，把整体的、概要性的功能要求进行层层分解与展开，分解成具体的、有相对独立性的基本的功能要求。这样就可以把被测试软件中所有的因子都确定下来，并为确定因子的权值提供参考的依据。确定因子与状态是设计测试用例的关键。因此，要求尽可能全面地、正确地确定取值，以确保测试用例的设计做到完整与有效。
- 加权筛选，生成因素分析表。对因子与状态的选择可按其重要程度分别加权。可根据各个因子及状态作用的大小、出现频率的大小以及测试的需要，确定权值的大小。
- 利用正交表构造测试数据集，正交表的推导依据 Galois 理论。

利用正交试验设计方法设计测试用例，与使用等价类划分、边界值分析、因果图等方法相比，有以下优点：节省测试工作工时；可控制生成的测试用例的数量；测试用例具有一定的覆盖率。

正交试验法在软件测试中是一种有效的方法，例如在平台参数配置方面，我们要选择哪种组合方式是最好的，每个参数可能就是一个因子，参数的不同取值就是水平，这样我们可以采用正交试验法设计出最少的测试组合，达到有效的测试目的。

5.2.8　功能图法

一个程序的功能说明通常由动态说明和静态说明组成。动态说明描述了输入数据的次序或转移的次序。静态说明描述了输入条件与输出条件之间的对应关系。对于较复杂的程序，由于存在大量的组合情况，因此，仅用静态说明组成的规格说明对于测试来说往往是不够的，必须用动态说明来补充功能说明。

1．功能图设计方法

功能图方法是用功能图形象地表示程序的功能说明，并机械地生成功能图的测试用例。功能图模型由状态迁移图和逻辑功能模型构成。

- 状态迁移图用于表示输入数据序列以及相应的输出数据。在状态迁移图中，由

输入数据和当前状态决定输出数据和后续状态。

- 逻辑功能模型用于表示在状态中输入条件和输出条件之间的对应关系。逻辑功能模型只适合于描述静态说明，输出数据仅由输入数据决定。测试用例则是由测试中经过的一系列状态和在每个状态中必须依靠输入/输出数据满足的一对条件组成。

功能图方法实际上是一种黑盒、白盒混合用例设计方法。

功能图方法中要用到逻辑覆盖和路径测试的概念和方法，属白盒测试方法中的内容。逻辑覆盖是以程序内部的逻辑结构为基础的测试用例设计方法，该方法要求测试人员对程序的逻辑结构有清楚的了解。由于覆盖测试的目标不同，逻辑覆盖可分为：语句覆盖、判定覆盖、判定-条件覆盖，条件组合覆盖及路径覆盖。下面我们指的逻辑覆盖和路径是功能或系统水平上的，以区别于白盒测试中的程序内部的，如图 5-9 及表 5-9 所示。

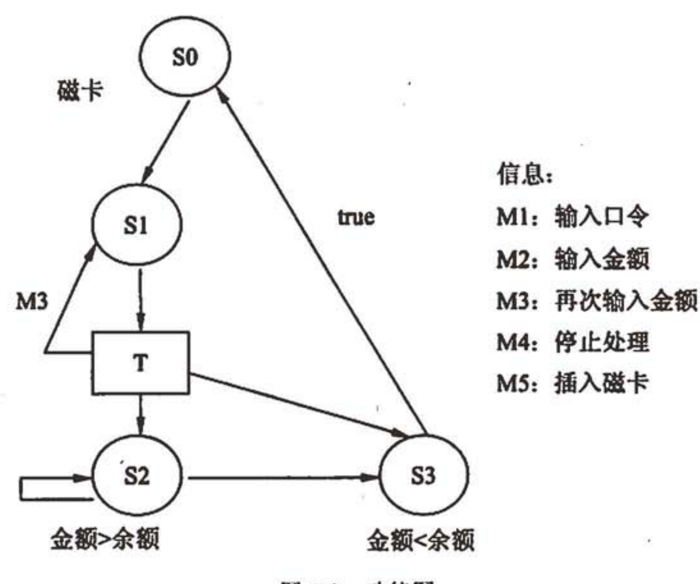


图 5-9 功能图

表 5-9 判 定 表

输 入	口令＝记录	Y	N	N
	错输入＝3 次	N	Y	N
输 出	M2	—		
	M3			—
	M4		—	
	消去卡		—	

续表

状　态	S1			—
	S2	—		
	S3		—	

2. 功能图法生成测试用例

功能图由状态迁移图和布尔函数组成。状态迁移图用状态和迁移来描述一个状态，指出数据输入的位置（或时间），而迁移则指明状态的改变，同时要依靠判定表和因果图表示的逻辑功能。

采用什么样的方法生成测试用例？从功能图生成测试用例，得到的测试用例数是可接受的。问题的关键是如何从状态迁移图中选取测试用例。若用节点代替状态，用弧线代替迁移，状态迁移图就可转化成一个程序的控制流程图形式。问题就转化为程序的路径测试问题（白盒测试范畴概念）了。

测试用例生成规则：为了把状态迁移（测试路径）的测试用例与逻辑模型的测试用例组合起来，从功能图生成实用的测试用例，需定义下面的规则。一个结构化的状态迁移中，定义 3 种形式的循环：顺序、选择和重复。但分辨一个状态迁移中的所有循环是有困难的。

从功能图生成测试用例的过程如下。

- 生成局部测试用例：在每个状态中，从因果图生成局部测试用例。局部测试库由原因值（输入数据）组合与对应的结果值（输出数据或状态）构成。
- 测试路径生成：利用上面的规则生成从初始状态到最后状态的测试路径。
- 测试用例合成：合成测试路径与功能图中每个状态的局部测试用例。结果是视状态到最后状态的一个状态序列，以及每个状态中输入数据与对应输出数据组合。
- 测试用例的合成算法：采用条件构造树。

5.2.9　场景法

现在的软件几乎都是用事件触发来控制流程的，事件触发时的情景便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这种在软件设计方面的思想也可引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

提出这种测试思想的是 Rational 公司，并在 RUP2000 中文版中有详尽的解释和应用。

用例场景用来描述流经用例的路径，从用例开始到结束遍历这条路径上所有基本流和备选流。



1．基本流和备选流

如图 5-10 所示，图中经过用例的每条路径都用基本流和备选流来表示，直黑线表示基本流，是经过用例的最简单的路径。备选流用不同的彩色表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中（如备选流 1 和 3）；也可能起源于另一个备选流（如备选流 2），或者终止用例而不再重新加入到某个流（如备选流 2 和 4）。

按照如图 5-10 中所示的每个经过用例的路径，可以确定以下不同的用例场景。

场景 1：基本流；

场景 2：基本流、备选流 1；

场景 3：基本流、备选流 1、备选流 2；

场景 4：基本流、备选流 3；

场景 5：基本流、备选流 3、备选流 1；

场景 6：基本流、备选流 3、备选流 1、备选流 2；

场景 7：基本流、备选流 4；

场景 8：基本流、备选流 3、备选流 4。

注：为方便起见，场景 5、6 和 8 只考虑了备选流 3 循环执行一次的情况。

需要说明的是，为了能清晰地说明场景，我们所举的例子都非常简单，在实际应用中，测试用例很少如此简单。

2．ATM 例子

（1）例子描述

如图 5-11 所示是 ATM 例子的流程示意图。

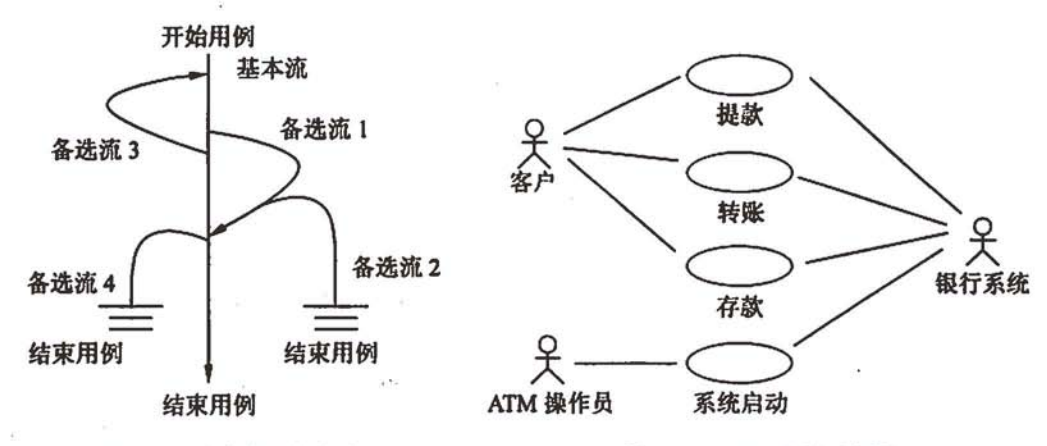


图 5-10 基本流和备选流　　　　图 5-11 ATM 流程示意图

如表 5-10 所示，包含了如图 5-11 中所示提款用例的基本流和某些备用流。

表 5-10　用　例　流

基本流	本用例的开始是 ATM 处于准备就绪状态 准备提款：客户将银行卡插入 ATM 机的读卡机 验证银行卡：ATM 机从银行卡的磁条中读取账户代码，并检查它是否属于可以接收的银行卡 输入 PIN：ATM 要求客户输入 PIN 码（4 位）验证账户代码和 PIN，验证账户代码和 PIN，以确定该账户是否有效以及所输入的 PIN 对该账户来说是否正确。对于此事件流，账户是有效的，而且 PIN 对此账户来说正确无误 ATM 选项：ATM 显示在本机上可用的各种选项。在此事件流中，银行客户通常选择“提款” 输入金额：要从 ATM 中提取的金额。对于此事件流，客户需选择预设的金额（10 元、20 元、50 元或 100 元）。授权 ATM 通过将卡 ID、PIN、金额以及账户信息作为一笔交易发送给银行系统来启动验证过程。对于此事件流，银行系统处于联机状态，而且对授权请求给予答复，批准完成提款过程，并且据此更新账户余额 出钞：提供现金 返回银行卡：银行卡被返还 收据：打印收据并提供给客户。ATM 还相应地更新内部记录 用例结束时 ATM 又回到准备就绪状态
备选流 1——银行卡无效	在基本流步骤 2 中验证银行卡，如果卡是无效的，则卡被退回，同时会通知相关消息
备选流 2——ATM 内没有现金	在基本流步骤 5 中 ATM 选项，选项将无法使用。如果 ATM 内没有现金，则“提款”选项不可用
备选流 3——ATM 内现金不足	在基本流步骤 6 中输入金额，如果 ATM 机内金额少于请求提取的金额，则将显示一则适当的消息，并且在步骤 6 输入金额处重新加入基本流
备选流 4——PIN 有误	在基本流步骤 4 中验证账户和 PIN，客户有三次机会输入 PIN。如果 PIN 输入有误，ATM 将显示适当的消息；如果还存在输入机会，则此事件流在步骤 3 输入 PIN 处重新加入基本流。如果最后一次尝试输入的 PIN 码仍然错误，则该卡将被 ATM 机保留，同时 ATM 返回到准备就绪状态，本用例终止
备选流 5——账户不存在	在基本流步骤 4 中验证账户和 PIN，如果银行系统返回的代码表明找不到该账户或禁止从该账户中提款，则 ATM 显示适当的消息并且在步骤 9 返回银行卡处重新加入基本流



续表

备选流 6——账面金额不足	在基本流步骤 7 授权中，银行系统返回代码表明账户余额少于在基本流步骤 6 输入金额内输入的金额，则 ATM 显示适当的消息并且在步骤 6 输入金额处重新加入基本流
备选流 7——达到每日最大的提款金额	在基本流步骤 7 授权中，银行系统返回的代码表明包括本提款请求在内，客户已经或将超过在 24 小时内允许提取的最多金额，则 ATM 显示适当的消息并在步骤 6 输入金额上重新加入基本流
备选流 x——记录错误	如果在基本流步骤 10 收据中，记录无法更新，则 ATM 进入"安全模式"，在此模式下所有功能都将暂停使用。同时向银行系统发送一条适当的警报信息表明 ATM 已经暂停工作
备选流 y——退出	客户可随时决定终止交易（退出）。交易终止，银行卡随之退出
备选流 z——"翘起"	ATM 包含大量的传感器，用以监控各种功能，如电源检测器、不同的门和出入口处的测压器以及动作检测器等。在任一时刻，如果某个传感器被激活，则警报信号将发送给警方而且 ATM 进入"安全模式"，在此模式下所有功能都暂停使用，直到采取适当的重启/重新初始化的措施
第一次迭代中，根据迭代计划，我们需要核实提款用例已经正确地实施。此时尚未实施整个用例，只实施了下面的事件流： 基本流——提取预设金额（10 元、20 元、50 元、100 元） 备选流 2——ATM 内没有现金 备选流 3——ATM 内现金不足 备选流 4——PIN 有误 备选流 5——账户不存在/账户类型有误 备选流 6——账面金额不足	

（2）场景设计

如表 5-11 所示是生成的场景。

表 5-11　场 景 设 计

场 景 描 述	基 本 流	备 选 流
场景 1——成功的提款	基本流	
场景 2——ATM 内没有现金	基本流	备选流 2
场景 3——ATM 内现金不足	基本流	备选流 3
场景 4——PIN 有误（还有输入机会）	基本流	备选流 4
场景 5——PIN 有误（不再有输入机会）	基本流	备选流 4
场景 6——账户不存在/账户类型有误	基本流	备选流
场景 7——账户余额不足	基本流	备选流

注：为方便起见，备选流 3 和 6（场景 3 和 7）内的循环以及循环组合未纳入表中。

（3）用例设计

对于这 7 个场景中的每一个场景都需要确定测试用例，一般采用矩阵或决策表来确定和管理测试用例。如表 5-12 所示是一种通用格式，其中行代表各个测试用例，列代表测试用例的信息。本例中的测试用例包含测试用例 ID、场景/条件、测试用例中涉及的所有数据元素和预期结果等项目。首先确定执行用例场景所需的数据元素，然后构建矩阵，最后要确定包含执行场景所需的适当条件的测试用例。在下面的矩阵中，V 表示这个条件必须是有效的才可执行基本流，I 表示这种条件下将激活所需备选流 ，n/a 表示这个条件不适用于测试用例。

表 5-12　测试用例表

TC（测试用例）ID 号	场景/条件	PIN	账号	输入（或选择）的金额	账面金额	ATM 内的金额	预期结果
CW1	场景 1——成功的提款	V	V	V	V	V	成功的提款
CW2	场景 2——ATM 内没有现金	V	V	V	V	I	提款选项不可用，用例结束
CW3	场景 3——ATM 内现金不足	V	V	V	V	I	警告消息，返回基本流步骤 6-输入金额
CW4	场景 4 ——PIN 有误（还有不止一次输入机会）	I	V	n/a	V	V	警告消息，返回基本流步骤 4，输入 PIN
CW5	场景 4——PIN 有误（还有一次输入机会）	I	V	n/a	V	V	警告消息，返回基本流步骤 4，输入 PIN
CW6	场景 4——PIN 有误（不再有输入机会）	I	V	n/a	V	V	警告消息，卡予保留，用例结束

在上面的矩阵中，六个测试用例执行了四个场景。对于基本流，上述测试用例 CW1 被称为正面测试用例。它一直沿着用例的基本流路径执行，未发生任何偏差。基本流的全面测试必须包括负面测试用例，以确保只有在符合条件的情况下才执行基本流。这些负面测试用例由 CW2～CW6 表示。虽然 CW2～CW6 相对于基本流而言都是负面测试用例，但它们相对于备选流 2～4 而言是正面测试用例。而且对于这些备选流中的每一个而言，至少存在一个负面测试用例，就是 CW1-基本流。

每个场景只有一个正面测试用例和负面测试用例是不充分的，场景 4 正是这样的一个示例。要全面地测试场景 4-PIN 有误，至少需要三个正面测试用例，以激活场景 4：

① 输入了错误的 PIN，但仍存在输入机会，此备选流重新加入基本流中的步骤



3-输入 PIN。

② 输入了错误的 PIN，而且不再有输入机会，则此备选流将保留银行卡并终止用例。

③ 最后一次输入时输入了“正确”的 PIN。备选流在步骤 5-输入金额处重新加入基本流。

注意，在上面的矩阵中，无需为条件输入任何实际的值。以这种方式创建测试用例矩阵的一个优点在于容易看到测试的是什么条件。由于只需要查看 V 和 I，这种方式还易于判断是否已经确定了充足的测试用例。从表 5-12 中可发现存在几个无效的条件 I，这表明测试用例还不完全，如场景 6-不存在的账户/账户类型有误和场景 7-账户余额不足就缺少测试用例。

（4）数据设计

一旦确定了所有的测试用例，则应对这些用例进行复审和验证以确保其准确且适度，并取消多余或等效的测试用例。

表 5-13 测试数据表

TC（测试用例）ID 号	场景/条件	PIN	账号	输入的金额（或选择的金额）	账面金额（元）	ATM 内的金额（元）	预期结果
CW1	场景 1——成功的提款	4987	809-498	50.00	500.00	2 000	成功的提款。账户余额被更新为 450.00
CW2	场景 2——ATM 内没有现金	4987	809-498	100.00	500.00	0.00	提款选项不可用，用例结束
CW3	场景 3——ATM 内现金不足	4987	809-498	100.00	500.00	70.00	警告消息，返回基本流步骤 6 输入金额
CW4	场景 4——PIN 有误（还有不止一次的输入机会）	4978	809-498	n/a	500.00	2 000	警告消息，返回基本流步骤 4，输入 PIN
CW5	场景 4——PIN 有误（还有一次输入机会）	4978	809-498	n/a	500.00	2 000	警告消息，返回基本流步骤 4，输入 PIN
CW6	场景 4——PIN 有误（不再有输入机会）	4978	809-498	n/a	500.00	2 000	警告消息，卡予保留，用例结束

测试用例一经认可，就可以确定实际数据值（在测试用例实施矩阵中）并且设定测试数据。

以上测试用例只是在本次迭代中需要用来验证提款用例的一部分测试用例。需要的其他测试用例包括以下内容。

场景 6——账户不存在/账户类型有误：未找到账户或账户不可用；

场景 6——账户不存在/账户类型有误：禁止从该账户中提款；

场景 7——账户余额不足：请求的金额超出账面金额。

在将来的迭代中，当实施其他事件流时，在下列情况下将需要测试用例：

① 无效卡（所持卡为挂失卡、被盗卡、非承兑银行发卡、磁条损坏等）；

② 无法读卡（读卡机堵塞、脱机或出现故障）；

③ 账户已消户、冻结或由于其他方面原因而无法使用；

④ ATM 内的现金不足或不能提供所请求的金额（与 CW3 不同，在 CW3 中只是一种币值不足，而不是所有币值都不足）；

⑤ 无法联系银行系统以获得认可；

⑥ 银行网络离线或交易过程中断电。

结论：所有从事软件测试和即将从事软件测试的人大都是从黑盒测试做起的，每种类型的软件有各自的特点，每种测试用例设计的方法也有各自的特点，针对不同软件如何利用这些黑盒方法是非常重要的，它能极大地提高测试效率和测试覆盖度，认真掌握这些方法的原理，有效提高测试水平，积累更多的测试经验，这是测试人员最宝贵的财富。

5.2.10　测试方法选择的综合策略

测试用例的设计方法不是单独存在的，具体到每个测试项目里都会用到多种方法，每种类型的软件有各自的特点，每种测试用例设计的方法也有各自的特点，针对不同软件如何利用这些黑盒方法是非常重要的，在实际测试中，往往是综合使用各种方法才能有效地提高测试效率和测试覆盖度，这就需要认真掌握这些方法的原理，积累更多的测试经验，以有效地提高测试水平。

以下是各种测试方法选择的综合策略，可供读者在实际应用过程中参考。

① 首先进行等价类划分，包括输入条件和输出条件的等价划分，将无限测试变成有限测试，这是减少工作量和提高测试效率最有效的方法。

② 在任何情况下都必须使用边界值分析方法。经验表明，用这种方法设计出的测试用例发现程序错误的能力最强。



③ 可以用错误推测法追加一些测试用例，这需要依靠测试工程师的智慧和经验。

④ 对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度。如果没有达到要求的覆盖标准，应当再补充足够的测试用例。

⑤ 如果程序的功能说明中含有输入条件的组合情况，则一开始就可选用因果图法和判定表驱动法。

⑥ 对于参数配置类的软件，要用正交试验法选择较少的组合方式达到最佳效果。

⑦ 功能图法也是很好的测试用例设计方法，我们可以通过不同时期条件的有效性设计不同的测试数据。

⑧ 对于业务流清晰的系统，可以利用场景法贯穿整个测试案例过程，在案例中综合使用各种测试方法。

5.3 测试用例的编写

5.3.1 测试用例计划的目的

仔细计划测试用例，是达成测试目标的必由之路。这样做的重要性体现如下。

即使在小型软件项目上，也可能有数千个测试用例。建立用例可能需要一些测试员经过几个月甚至几年的时间。正确的计划会组织好用例，以便全体测试员和其他项目小组成员有效地审查和使用。

我们已经知道，在项目期间有必要多次执行同样的测试，以寻找新的软件缺陷，保证老的软件缺陷得以修复。假如没有正确的计划，就不可能知道需要执行哪个测试用例，原有的测试是否得到重复。

有时我们需要回答整个项目期间的重要问题。例如，计划执行多少个测试用例；在软件最终版本上执行多少个测试用例；多少个通过，多少个失败；有忽略的测试用例吗，等等。如果没有测试用例计划，就不能回答这些问题。

在少数高风险行业中，软件测试小组必须证明确实执行了计划执行的测试。发布忽略某些测试用例的软件是危险的。正确的测试用例计划和跟踪提供了一种证实测试的手段。

5.3.2 测试设计说明

大家知道，项目整体测试计划的级别非常高。它虽然把软件拆分为具体特性和可测试项，并将其分派到每个测试员头上，但是它并没有指明如何对这些特性进行测试，可能仅仅对使用自动化测试还是黑盒测试或者白盒测试有一些提示，但是并不会涉及如何

使用以及在哪里使用这些工具。

为了更好地进行测试，我们需要为单个软件特性定义具体的测试方法，这就是测试设计说明。ANSI/IEEE 829 中对测试设计说明的解释是：测试设计说明就是在测试计划中提炼测试方法，要明确指出设计包含的特性以及相关的测试用例和测试程序，并指定判断特性通过/失败的规则。

测试设计说明的目的是组织和描述针对具体特性需要进行的测试。然而，它并不给出具体的测试用例或者执行测试的步骤。以下内容来自于 ANSI/IEEE 829 标准，应该作为测试设计说明的部分内容。

- 标识符：用于引用和定位测试设计说明的惟一标识符。该说明还应该引用整个测试计划，还应该包含任何其他计划或者说明的引用。
- 要测试的特性：对测试设计说明所包含的软件特性的描述。例如"计算器程序的加法功能"、"写字板程序中的字体大小选择和显示"、"QuickTime 软件的视频卡配置测试"。该部分还将明确指出要间接测试的特性，它通常作为主特性的辅助特性。例如，文件打开对话框的用户界面虽然不在测试设计说明中重点指出，但是在测试读写功能的过程中要间接测试。
- 方法：描述测试的通用方法。如果方法在测试计划中列出，就应该在此详细描述要使用的技术，并给出如何验证测试结果的方法。例如，我们这样描述一种方法，开发一种测试工具，顺序读写不同大小的数据文件，数据文件的数目和大小及包含的内容由程序员提供的示例来确定。用文件比较工具比较输出的文件和源文件，如果相同，则认为通过；如果不同，则认为失败。
- 测试用例信息：用于描述所引用的测试用例的相关信息。应该列出所选的等价区间，给出测试用例的引用信息以及用于执行测试用例的测试程序说明。例如："检查最大值　测试用例 ID#15326"、"检查最小值　测试用例 ID#15327"，在这部分不定义实际测试用例。
- 通过/失败规则：描述用什么规则来判定某项特性的测试结果是通过还是失败。这种描述有可能非常简单和明确，例如"通过是指当执行全部测试用例时没有发现软件缺陷"。也有可能不是非常明确，例如"失败是指 10%以上的测试用例没有通过"。

5.3.3　测试用例说明

如何记录和记载创建的测试用例？如果你已经开始进行一些软件测试了，就可能采用过一些用例描述格式。本节讲解编写测试用例的有关内容，指出将要考虑的重点。

ANSI/IEEE 829 标准称测试用例说明为编写用于输入输出的实际数值和预期结果，



同时还明确指出，使用具体测试用例产生的测试程序的限制。一个测试用例的编写可参考表 5-14。

表 5-14 测 试 用 例

编号：

编制人		审定人		时间	
软件名称			编号/版本		
测试用例					
用例编号					
参考信息（参考的文档及章节号或功能项）：					
输入说明（列出选用的输入项，覆盖正常、异常情况）：					
输出说明（逐条与输入项对应，列出预期输出）：					
环境要求（测试要求的软、硬件、网络要求）：					
特殊规程要求：					
用例间的依赖关系：					
用例产生的测试程序限制：					

测试用例应该解释要向软件发送什么值或者条件，以及预期结果。一个测试用例说明可以由多个测试用例说明来引用，也可以引用多个测试程序。ANSI/IEEE 829 标准还列出了一些应该包含在内的重要信息，如下所示。

- 标识符：由测试设计过程说明和测试程序说明引用的惟一标识符。
- 测试项：描述被测试的详细特性、代码模块等，应该比测试设计说明中所列的特性更加具体。如果测试设计说明提到“计算器程序的加法功能”，那么测试用

例说明就会相应地提到“加法运算的上限溢出处理”。它还要指出引用的产品说明书或者测试用例所依据的其他设计文档。

- 输入说明：该说明列举执行测试用例的所有输入内容或者条件。如果测试计算器程序，输入说明可能简单到“1+1”。如果测试蜂窝电话交换软件，输入说明可能是成百上千种输入条件。如果测试基于文件的产品，输入说明可能是文件名和内容的描述。
- 输出说明：描述进行测试用例预期的结果。例如，1+1 等于 2 吗？在蜂窝软件中上千个输出变量设置正确吗？读取文件的全部内容和预想的一样吗？
- 环境要求：是指执行测试用例必要的硬件、软件、测试工具、人员等。
- 特殊要求：描述执行测试必须的特殊要求。测试写字板程序也许不需要任何特殊条件，但是测试一些特殊的软件（如核电站软件）就有特殊要求。
- 用例之间的依赖性：如果一个测试用例依赖于其他用例，或者受其他用例的影响，就应该在此注明。

如果按照这里推荐的文档格式，对于每一个测试用例至少都要写上一页的描述文字，数千个测试用例可能要形成几千页文档。所以我们经常把 ANSI/IEEE 829 标准当作规范而不是标准使用（除非必须这样做，许多政府项目和某些行业要求按照此规格编写测试用例，但是在大多数情况下可以采用简便方法）。

采用简便方法并不是说放弃或者忽视重要的信息，而是意在找出一个更有效的方法对这些信息进行精简，例如，没有必要刻意要求不能用书面段落形式表述测试用例。如表 5-15 给出了一个打印机兼容性简单列表的例子。

表 5-15　打印机兼容性简单列表

测试用例序列号	型　号	模　式	黑　白	选　项
WP0001	Canon	BJC-7000	黑白	文字
WP0002	Canon	BJC-7000	黑白	超级照片
WP0003	Canon	BJC-7000	黑白	自动
WP0004	Canon	BJC-7000	黑白	草稿
WP0005	Canon	BJC-7000	彩色	文字
WP0006	Canon	BJC-7000	彩色	超级照片
WP0007	Canon	BJC-7000	彩色	自动
WP0008	Canon	BJC-7000	彩色	草稿
WP0009	HP	LaserJet Ⅳ	高	
WP0010	HP	LaserJet Ⅳ	中	
WP0011	HP	LaserJet Ⅳ	低	



表中的每一行是一个测试用例，有自己的标识符。伴随测试用例的所有其他信息，例如测试项、输入说明、输出说明、环境要求、特殊要求和依赖性等对所有这些用例都必须有，可以一并编写，附加到表格中。审查测试用例的人可以快速看完测试用例信息，然后审查表格，检查其范围。

表述测试用例的其他选择有大纲、状态表或数据流程图等方式。

5.3.4 测试程序说明

编写完测试设计和测试用例之后，就要说明执行测试用例的程序。什么是测试程序呢？ANSI/IEEE 829 标准把测试程序定义为“明确指出为实现相关测试设计而执行具体测试用例和操作软件系统的全部步骤”。

测试程序，有时也叫“测试脚本说明”，详细定义了执行测试用例的每一步操作。以下是需要定义的内容。

- 标识符：用来把测试程序与相关测试用例和测试设计相联系的惟一标识。
- 目的：本程序描述的目的以及将要执行的测试用例的引用信息。
- 特殊要求：执行测试所需的其他程序、特殊测试技术或者特殊设备。
- 程序步骤：执行测试用例的详细描述。它包含以下内容。

① 日志：指出用什么方法记录测试结果和现象。

② 设置：说明如何准备测试。

③ 启动：说明启动测试的步骤。

④ 程序：描述运行测试的步骤。

⑤ 衡量标准：描述如何判断结果。

⑥ 关闭：描述因意外原因而推迟测试的步骤。

⑦ 终止：描述正常停止测试的步骤。

⑧ 重置：说明如何把环境恢复到测试前的状态。

⑨ 偶然事件：说明如何处理计划之外的情况。

如果我们把测试程序只理解成“尝试执行所有的测试用例并报告发现的问题”是不够的。这虽然简单、容易，但是无法告诉新加入的测试员如何进行测试，不能重复而且无法证明哪些步骤执行了。使用详细的程序说明，则把要测试什么、如何测试等问题都表述得一目了然。如图 5-12 所示是“Windows 计算器”的测试程序说明的例子片断。

标识号：计算器。
目的：本程序说明描述执行加法测试用例的步骤。
特殊要求：本次测试不需要特殊的硬件和软件。
程序步骤：
日志：测试员按测试要求记录程序执行过程，所有必须填写的项都必须填写，包括问题的记录。
设置：测试者必须安装 Windows 98 的干净副本，使用测试工具 Tool-A 和 Tool-B 等。
启动：启动 Windows 98，单击开始按钮，选择程序，选择附件、选择计算器。
程序：用键盘输入每个测试用例并比较结果。
衡量标准：……

图 5-12　测试程序说明片断

5.3.5　测试用例细节探讨

俗话说“做什么都要适可而止”，测试用例计划也一样。测试用例计划包括四个目标，即组织性、重复性、跟踪和测试证实。开发测试用例的软件测试工程师要力争实现这些目标，但是其实现程度取决于行业、公司、项目和测试组的具体情况，通常也不太可能按照最细致的程度去编写测试用例。

我们设计的测试用例计划要力求达到最佳的详细程度，比如，在一个测试程序中要求在 PC 机上安装 Windows 2000 来执行测试，测试程序在其设置部分声明需要 Windows 2000，但是未声明 Windows 2000 的哪个版本。那么一两年内出现新版本会怎样？测试程序需要升级来反映这个变化吗？为了避免这个问题，可以省略具体的版本，而用“可用的最新版本”这样的说明来替代。

无比详细的测试用例说明减少了测试的随意性，使测试可以很好地重复，使得无经验的测试人员按照测试用例说明也能执行测试。但是编写如此细致的测试用例说明要花费相当多的时间和精力，并且由于细节繁多，也会阻滞测试工作，造成测试执行时间变长。

开始编写测试用例时，最好是采用当前项目的标准，同时需要根据 ANSI/IEEE 829 标准定义的格式，看什么符合项目要求，并可以做适当的调整。

不同的测试工程师设计的测试用例也会有所不同。通常有经验的测试工程师设计出



来的测试用例，在深度及广度上会比经验少的测试工程师的完整，这也是所谓的测试经验值。举例来讲，客户反应前一版 V1.3 的软件在 Windows 98 的环境下运行时，在屏幕保护程序激活后会产生问题，开发工程师将这问题解决并且已提交修正版本供客户网络下载，并且目前开发工程师所开发的软件最新版本为 V1.5 版，软件测试工程师就必须在 V1.5 版的测试用例内，加入屏幕保护程序激活测试用例，甚至将这个用例增加至其他的测试平台。

第 6 章　白盒测试技术

6.1　白盒测试基本技术

6.1.1　词法分析与语法分析

通过词法分析与语法分析可以获取软件组成的重要基本因数，例如：变量标识符、过程标识符、常量等，组合这些基本因数可以得到软件的基本信息。如：

- 标号交叉引用表。列出在各模块中出现的全部标号，在表中标出标号的属性，包括已说明、未说明、已使用、未使用。表中还包括在模块以外的全局标号、计算标号等。
- 变量交叉引用表，即变量定义与引用表。在表中应标明各变量的属性，包括已说明、未说明、隐式说明以及类型及使用情况，进一步还可区分是否出现在赋值语句的右边，是否属于 COMMON 变量、全局变量或特权变量等。
- 子程序、宏和函数表。在表中列出各个子程序、宏和函数的属性，包括已定义、未定义、定义类型；以及参数表、输入参数的个数、顺序、类型，输出参数的个数、顺序、类型；已引用、未引用、引用次数等。
- 等价表。表中列出在等价语句或等值语句中出现的全部变量和标号。
- 常数表。表中列出全部数字常数和字符常数，并指出它们在哪些语句中首先被定义。

按功能分类，引用表的作用有以下 3 种。

- 直接从表中查出说明/使用错误。如循环层次表、变量交叉引用表、标号交叉引用表。
- 为用户提供辅助信息。如子程序（宏、函数）引用表、等价表、常数表。
- 用来做错误预测和程序复杂度计算。如操作符和操作数的统计表等。

6.1.2　静态错误分析

静态错误分析用于确定在源程序中是否有某类错误或“危险”结构。它有以下几种。

1．类型和单位分析

为了强化对源程序中数据类型的检查，在程序设计语言中扩充一些新的数据类型，

例如，仅能在数组中使用的"下标"类型及在循环语句中当作控制变量使用的"计数器"类型。这样就可以静态预处理程序，分析程序中的类型错误。

2．引用分析

在静态错误分析中，最广泛使用的技术就是发现引用异常。如果沿着程序的控制路径，变量在赋值以前被引用，或变量在赋值以后未被引用，这时就发生了引用异常。

为了检测引用异常，需要检查通过程序的每一条路径。通常采用类似深度优先的方法遍历程序流图的每一条路径，也可以建立引用异常的探测工具，这种工具包括两个表：定义表和未引用表。每张表中都包含一组变量表。未引用表中包括已被赋值但还未被引用的一些变量。

当扫描抵达一个出度大于1的节点V时，深度优先探测算法要求先检查最左分支的那一部分程序流图，然后再检查其他分支。在最左分支检查完之前，应把定义表与未引用表的当前内容用一个栈暂时存储起来，当最左分支检查完之后，算法控制返回到节点V，从栈中恢复该节点的定义表和未引用表的老的副表，然后再去遍历该节点的下一个分支，这个过程要继续到全部分支检查完为止。

3．表达式分析

对表达式进行分析，以发现和纠正在表达式中出现的错误。包括：

- 在表达式中不正确地使用了括号造成错误；
- 数组下标越界造成错误；
- 除数为零造成错误；
- 对负数开平方，或对π求正切造成错误。

最复杂的一类表达式分析是对浮点数计算的误差进行检查。由于使用二进制数不精确地表示十进制浮点数，常常使计算结果出乎意料。

4．接口分析

接口一致性是程序的静态错误分析和设计分析共同研究的题目。接口一致性的设计可以分析检查模块之间接口的一致性和模块与外部数据库之间接口的一致性。

程序关于接口的静态错误分析检查过程与实参在类型、函数过程之间接口的一致性，因此要检查形参与实参在类型、数量、维数、顺序、使用上的一致性；检查全局变量和公共数据区在使用上的一致性。

6.1.3 程序插桩技术

在软件动态测试中，程序插桩（Program Instrumentation）是一种基本的测试手段，有着广泛的应用。

1．方法简介

程序插桩方法，简单地说，是借助往被测程序中插入操作，来实现测试目的的方法。

我们在调试程序时，常常要在程序中插入一些打印语句。其目的在于，希望执行程序时，打印出我们最为关心的信息。进一步通过这些信息了解执行过程中程序的一些动态特性。比如，程序的实际执行路径，或是特定变量在特定时刻的取值。从这一思想发展出的程序插桩技术能够按用户的要求，获取程序的各种信息，成为测试工作的有效手段。

如果我们想要了解一个程序在某次运行中所有可执行语句被覆盖（或称被经历）的情况，或是每个语句的实际执行次数，最好的办法是利用插桩技术。这里仅以计算整数 X 和整数 Y 的最大公约数程序为例，说明插桩方法的要点。图 6-1 所示给出了这一程序的流程图。图中虚线框并不是源程序的内容，而是为了记录语句执行次数而插入的。

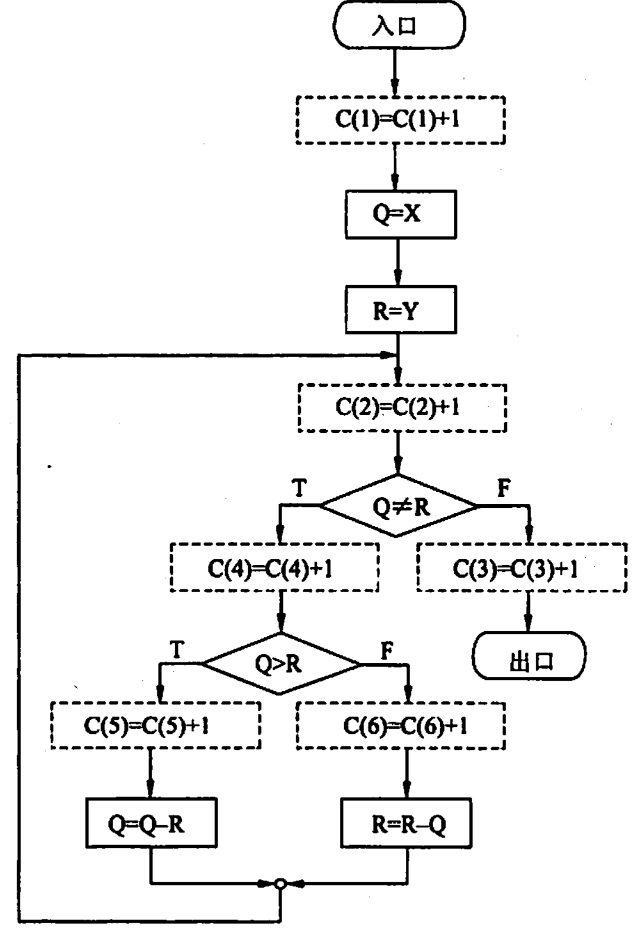


图 6-1　插桩后求最大公约数程序的流程图

这些虚线框要完成的操作都是计数语句，其形式为：

$$C(i) = C(i) + 1 \qquad i = 1, 2, \ldots, 6$$

程序从入口开始执行，到出口结束。凡经历的计数语句都能记录下该程序点的执行次数。如果我们在程序的入口处还插入了对计数器 C（i）初始化的语句，在出口处插入了打印这些计数器的语句，就构成了完整的插桩程序，它便能记录并输出在各程序点上语句的实际执行次数。如图 6-2 所示表示了插桩后的程序，图中箭头所指均为插入的语句（源程序语句略）。

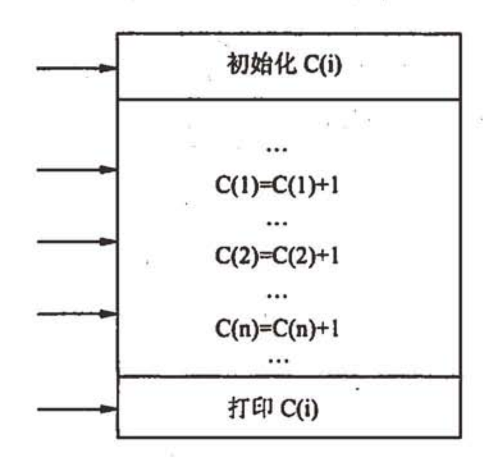


图 6-2 插桩程序中插入的语句

通过插入的语句获取程序执行中的动态信息，这一做法如同在刚研制成的机器特定部位安装记录仪表一样。安装好以后开动机器试运行，我们除了可以从机器加工的成品检验得知机器的运行特性外，还可通过记录仪表了解其动态特性。这就相当于在运行程序以后，一方面可检验测试的结果数据，另一方面还可借助插入语句给出的信息了解程序的执行特性。正是这个原因，有时把插入的语句称为“探测器”，借以实现“探查”或“监控”的功能。

在程序的特定部位插入记录动态特性的语句，最终是为了把程序执行过程中发生的一些重要历史事件记录下来。例如，记录在程序执行过程中某些变量值的变化情况，变化的范围等。又如本书前面章节中所讨论的程序逻辑覆盖情况，也只有通过程序的插桩才能取得覆盖信息。实践表明，程序插桩方法是应用很广的技术，特别是在完成程序的测试和调试时非常有效。

设计插桩程序时需要考虑的问题包括：

- 探测哪些信息；

- 在程序的什么部位设置探测点；
- 需要设置多少个探测点。

其中前两个问题需要结合具体情况解决，并不能给出笼统的回答。至于第三个问题，需要考虑如何设置最少探测点的方案。例如，在如图 6-1 所示的程序入口处，若要记录语句 Q = X 和 R = Y 的执行次数，只需插入 C(1) = C(1) + 1 这样一个计数语句就够了，没有必要在每个语句之后都插入一个计数语句。在一般的情况下，我们可以认为，在没有分支的程序段中只需一个计数语句。但程序中如果出现了多种控制结构，使得整个结构十分复杂，则为了在程序中设计最少的计数语句，需要针对程序的控制结构进行具体的分析。这里我们以 Fortran 程序为例，列举至少应在哪些部位设置计数语句：

- 程序块的第一个可执行语句之前；
- entry 语句的前后；
- 有标号的可执行语句处；
- do、do while、do until 及 do 终端语句之后；
- block-if、else if、else 及 endif 语句之后；
- logical if 语句处；
- 输入/输出语句之后；
- call 语句之后；
- 计算 go to 语句之后。

2．断言语句

在程序中的特定部位插入某些用以判断变量特性的语句，使得程序执行中这些语句得以证实，从而使程序的运行特性得到证实。我们把插入的这些语句称为断言(assertions)。这一作法是程序正确性证明的基本步骤，尽管算不上严格的证明，但方法本身仍然是很实用的。下面以求两个非负数 NUM 和 DEN 之商的 Wensley 迭代算法为例，对断言语句的作用作一简要说明。

假定两个非负数中，NUM 小于 M（即所得之商小于 1），算法中只用到加、减及除 2 的运算。该迭代算法的程序如图 6-3 所示。

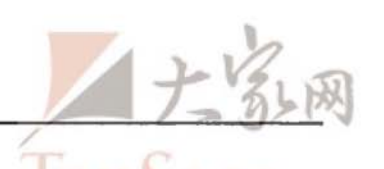
从程序中可以看出，在每次迭代中由分母得到的变量 B 以及权增量 W 都要缩小一半，而且变量 A 随着迭代次数的增加将接近分子。这些粗略的观察和分析可以用以下 4 个断言语句表达，在每次迭代开始时 4 个断言必定为真：

① $W = 2^{-K}$（K 是迭代次数，并且 $K \geqslant 0$）；

② A = DEN * Q；

③ B = DEN * W/2；

④ $NUM/DEN - W < Q \leqslant NUM/DEN$



```
procedure DIVIDE(NUM, DEN, E, Q)
* E is the accuracy required. E≥0. Q is both      *
* the result at exit and at any interim stage.     *
* A. B and W are the other elements of the pro-*
* gram vector.
    Q:=0
    A:=0
    B:=DEN/2
    W:=1
    until W < E loop
        if (NUM-A-B) ≥ 0
        then
            Q:=Q + W/2
            A:=A + B
        endif
        B:=B/2
        W:=W/2
    endloop
    end
```

图 6-3 计算两非负数之商的迭代程序

此外，我们还看出，在循环外 W < E，而且结果 Q 总是从下面逼近真正的商。这就得到了输出断言：

$$NUM/DEN - E < Q \leq NUM/DEN$$

它和上面的第④断言很相似。

假定我们所用的编译系统能够处理表达式形式的断言语句，插入断言以后的程序如图 6-4 所示。其中带有标记@的语句是断言语句。新增加的变量 K 只是在计算第①断言时用到。

首先来检验在初始化以后循环后的断言。

① 由于 K = 0，所以是初值。W = 2–K = 1。

② 由于 Q = 0，A 的初始 A = DEN * Q = 0。

③ 将 W 的值代入 DEN * W/2，则得 B 的初值：B = DEN/2。

④ 我们曾假定 0 ≤ NUM < DEN，NUM/DEN – 1（W 的值）必定小于零，因而也就小于 Q（它是零）。而且，NUM/DEN 必定大于 Q，因为 NUM 和 DEN 均为正，

```
  procedure DIVIDE(NUM, DEN, E, Q)
* E is the required accuracy. E≥0. Q is both      *
* the result at exit and at any interim stage.    *
* A. B and W are the other elements of the pro-*
* gram vector.
    Q:=0
    A:=0
    B:=DEN/2
    W:=1
@   K:=0
    until K< E loop
@   assert W=1/2**K
@   assert A=DEN*Q
@   assert B=DEN*W/2
@   assert NUM/DEN-W<Q and Q≤ NUM/DEN
    if (NUM-A-B)≥0
    then
        Q:=Q+W/2
        A:=A+B
    endif
    B:=B/2
    W:=W/2
@   K:=K+1
    endloop
@   assert NUM/DEN-E<Q and Q≤NUM/DEN
    end
```

图 6-4　插入断言后的迭代程序

Q 为零。

以上说明了这些断言在初始状态下为真。如果继续迭代，要证明断言为真，就必须证明无论 if-then 结构中执行什么路径这些断言都为真。我们先来考虑在初始测试中 $NUM-A-B \geq 0$ 为假，即检验失败，然后给出程序向量的新值（A′, B′, W′, Q′和 K′），我们有：

$$A^{*} = A$$
$$B^{*} = B/2$$



$$W^* = W/2$$
$$Q^* = Q$$
$$K^* = K + 1$$

再来检验 4 个断言：

① $W^* = W/2 = 1/2 ** K^*$

② $A^* = A = DEN * Q = DEN * Q^*$

③ $B^* = B/2 = DEN * W/4 = DEN * W^*/2$

④ 把 A 和 B 代入（$NUM - A - B < 0$），得到 $NUM - DEN * Q - DEN * W/2 < 0$，对此关系式两端除以 DEN，并加 Q，得到，由于 $Q' = Q$，$W' = W/2$，我们有 $NUM/DEN - W^* < Q^*$，且 $Q \leqslant NUM/DEN$。

如果 if-then 检验成立，再来看 4 个断言。使用 A″，B″，W″，Q″和 K″作为新的程序向量，我们有：

① $W' = W/2 = 1/2 ** K''$

② $A'' = DEN * Q + DEN * W/2 = DEN * (Q + W/2) = DEN * Q''$

③ $B'' = B/2 = DEN * W/4 = DEN * W''/2$

④ 代入（$NUM - A - B \geqslant 0$），并作同前的变换，得到 $NUM/DEN - W''/2 < Q'' + W''/2$ 或：$NUM/DEN - W'' < Q''$，并且 $Q'' \leqslant NUM/DEN$。

总之，无论执行哪一路径，在每一迭代的开始，4 个断言均为真。尽管并未考虑输出断言，但是我们知道，第④断言成立，由于 $W < E$，$NUM/DEN - E < Q$ 和 $Q \leqslant NUM/DEN$ 必定为真，也就必定满足输出断言。

6.2 白盒测试方法

6.2.1 代码检查法

代码检查包括桌面检查、代码审查和走查等，主要检查代码和设计的一致性，代码对标准的遵循、可读性，代码逻辑表达的正确性，代码结构的合理性等方面；发现违背程序编写标准的问题，程序中不安全、不明确和模糊的部分，找出程序中不可移植部分、违背程序编程风格的问题，包括变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等内容。

1．代码检查方式

（1）桌面检查

这是一种传统的检查方法，由程序员检查自己编写的程序。程序员在程序通过编

译之后，对源程序代码进行分析、检验，并补充相关的文档，目的是发现程序中的错误。

由于程序员熟悉自己的程序及其程序设计风格，桌面检查由程序员自己进行可以节省很多的检查时间，但应避免主观片面性。

（2）代码审查

代码审查是由若干程序员和测试员组成一个审查小组，通过阅读、讨论和争议，对程序进行静态分析的过程。

代码审查分两步：第一步，小组负责人提前把设计规格说明书、控制流程图、程序文本及有关要求、规范等分发给小组成员，作为审查的依据。小组成员在充分阅读这些材料后，进入审查的第二步，召开程序审查会。在会上，首先由程序员逐句讲解程序的逻辑。在此过程中，程序员或其他小组成员可以提出问题，展开讨论，审查错误是否存在。实践表明，程序员在讲解过程中能发现许多原来自己没有发现的错误，而讨论和争议则促进了问题的暴露。例如，对某个局部性小问题修改方法的讨论，可能发现与之牵连的其他问题，甚至涉及到模块的功能说明、模块间接口和系统总体结构的大问题，从而导致对需求的重定义、重设计和重验证，进而大大改善了软件质量。

在会前，应当给审查小组每个成员准备一份常见错误的清单，把以往所有可能发生的常见错误罗列出来，供与会者对照检查，以提高审查的实效。

这个常见错误清单也称为检查表，它把程序中可能发生的各种错误进行分类，对每一类列举出尽可能多的典型错误，然后把它们制成表格，供再审查时使用。

（3）走查

走查与代码审查基本相同，其过程分为两步。

第一步也是把材料先发给走查小组每个成员，让他们认真研究程序，然后再开会。开会的程序与代码审查不同，不是简单地读程序和对照错误检查表进行检查，而是让与会者“充当”计算机，即首先由测试组成员为所测程序准备一批有代表性的测试用例，提交给走查小组。走查小组开会，集体扮演计算机角色，让测试用例沿程序的逻辑运行一遍，随时记录程序的踪迹，供分析和讨论用。

人们借助测试用例的媒介作用，对程序的逻辑和功能提出各种疑问，结合问题开展热烈的讨论和争议，能够发现更多的问题。

代码检查应在编译和动态测试之前进行，在检查前，应准备好需求描述文档、程序设计文档、程序的源代码清单、代码编码标准和代码缺陷检查表等。

在实际使用中，代码检查能快速找到缺陷，发现30%～70%的逻辑设计和编码缺陷，而且代码检查看到的是问题本身而非征兆。但是代码检查非常耗费时间，而且代码检查需要知识和经验的积累。



代码检查可以使用测试软件进行自动化测试，以利于提高测试效率，降低劳动强度，或者使用人工进行测试，以充分发挥人力的逻辑思维能力。

2．代码检查项目

- 检查变量的交叉引用表：重点是检查未说明的变量和违反了类型规定的变量；还要对照源程序，逐个检查变量的引用、变量的使用序列、临时变量在某条路径上的重写情况，局部变量、全局变量与特权变量的使用。
- 检查标号的交叉引用表：验证所有标号的正确性，检查所有标号的命名是否正确，转向指定位置的标号是否正确。
- 检查子程序、宏、函数：验证每次调用与所调用位置是否正确，确认每次所调用的子程序、宏、函数是否存在，检验调用序列中调用方式与参数顺序、个数、类型上的一致性。
- 等价性检查：检查全部等价变量的类型的一致性，解释所包含的类型差异。
- 常量检查：确认常量的取值和数制、数据类型，检查常量每次引用同它的取值、数制和类型的一致性。
- 标准检查：用标准检查工具软件或手工检查程序中违反标准的问题。
- 风格检查：检查发现程序在设计风格方面的问题。
- 比较控制流：比较由程序员设计的控制流图和由实际程序生成的控制流图，寻找和解释每个差异，修改文档并修正错误。
- 选择、激活路径：在程序员设计的控制流图上选择路径，再到实际的控制流图上激活这条路径。如果选择的路径在实际控制流图上不能被激活，则源程序可能有错。
- 对照程序的规格说明，详细阅读源代码，逐字逐句进行分析和思考，比较实际的代码和期望的代码，从它们的差异中发现程序的问题和错误。
- 补充文档：桌面检查的文档是一种过渡性的文档，不是公开的正式文档。通过编写文档，也是对程序的一种下意识的检查和测试，可以帮助程序员发现和抓住更多的错误。管理部门也可以通过审查桌面检查文档，了解模块的质量、完全性、测试方法和程序员的能力。

根据检查项目可以编制代码规则、规范和检查表等作为测试用例，如编码规范、代码检查规则、缺陷检查表等。

3．编码规范

编码规范是程序编写过程中必须遵循的规则，一般会详细规定代码的语法规则、语法格式等，如表 6-1 所示。

表 6-1　编码规范

序　号	总 则 条 款
	排版
1	1-1：程序块要采用缩进风格编写，缩进的空格数为 4 个
2	1-2：相对独立的程序块之间、变量说明之后必须加空行
3	1-3：较长的语句（>80 字符）要分成多行书写，长表达式要在低优先级操作符处划分新行，操作符放在新行之首，划分出的新行要进行适当的缩进，使排版整齐，语句可读
4	1-4：循环、判断等语句中若有较长的表达式或语句，则要进行适当的划分，长表达式要在低优先级操作符处划分新行，操作符放在新行之首
5	1-5：若函数或过程中的参数较长，则要进行适当的划分
6	1-6：不允许把多个短语句写在一行中，即一行只写一条语句
7	1-7：if、while、for、default、do 等语句自占一行
8	1-8：对齐只使用空格键，不使用 TAB 键
9	1-9：函数或过程的开始、结构的定义及循环、判断等语句中的代码都要采用缩进风格，case 语句下的情况处理语句也要遵从语句缩进要求
10	1-10：程序块的分界符（如 C/C++语言的大括号“{”和“}”）应各自独占一行并且位于同一列，同时与引用它们的语句左对齐。在函数体的开始、类的定义、结构的定义、枚举的定义以及 if、for、do、while0、switch、case 语句中的程序都要采用如上的缩进方式
11	1-11：在两个以上的关键字、变量、常量进行对等操作时，它们之间的操作符之前、之后或者前后要加空格；进行非对等操作时，如果是关系密切的，立即操作符（如—>），后不应加空格
	注释
1	2-1：一般情况下，源程序有效注释量必须在 20%以上
2	2-2：说明性文件（如头文件.h 文件、.inc 文件、.def 文件、编译说明文件.cfg 等）头部应进行注释，注释必须列出：版权说明、版本号、生成日期、作者、内容、功能、与其他文件的关系、修改日志等，头文件的注释中还应有函数功能简要说明
3	2-3：源文件头部应进行注释，列出版权说明、版本号、生成日期、作者、模块目的/功能、主要函数及其功能、修改日志等
4	2-4：函数头部应进行注释，列出函数的目的/功能、输入参数、输出参数、返回值、调用关系（函数、表）等
5	2-5：边写代码边注释，修改代码的同时修改相应的注释，以保证注释与代码的一致性。不再有用的注释要删除
6	2-6：注释的内容要清楚、明了，含义准确，防止注释二义性
7	2-7：避免在注释中使用缩写，特别是非常用缩写
8	2-8：注释应与其描述的代码相近，对代码的注释应放在其上方或右方（对单条语句的注释）相邻位置，不可放在下面，如放于上方则需与其上面的代码用空行隔开
9	2-9：对于所有有物理含义的变量、常量，如果其命名不是充分自注释的，在声明时都必须加以注释，说明其物理含义。变量、常量、宏的注释应放在其上方相邻位置或右方



续表

序 号	总 则 条 款
	注释
10	2-10：数据结构声明（包括数组、结构、类、枚举等），如果其命名不是充分自注释的，必须加以注释。对数据结构的注释应放在其上方相邻位置，不可放在下面；对结构中每个域的注释放在此域的右方
11	2-11：全局变量要有较详细的注释，包括对其功能、取值范围、哪些函数或过程存取它以及存取时的注意事项等的说明
12	2-12：注释与所描述内容进行同样的缩排
13	2-13：将注释与其上面的代码用空行隔开
14	2-14：对变量的定义和分支语句（条件分支、循环语句等）必须编写注释
15	2-15：对于 switch 语句下的 case 语句，如果因为特殊情况需要处理完一个 case 后进入下一个 case 处理，必须在该 case 语句处理完、下一个 case 语句前加上明确的注释
	标识符命名
1	3-1：标识符的命名要清晰、明了，有明确含义，同时使用完整的单词或大家基本可以理解的缩写，避免使人产生误解
2	3-2：命名中若使用特殊约定或缩写，则要有注释说明
3	3-3：自己特有的命名风格，要自始至终保持一致，不可来回变化
4	3-4：对于变量命名，禁止取单个字符（如 i、j、k…），建议除了要有具体含义外，还能表明其变量类型、数据类型等，但 i、j、k 作局部循环变量是允许的
5	3-5：命名规范必须与所使用的系统风格保持一致，并在同一项目中统一，比如采用 UNIX 的全小写加下划线的风格或大小写混排的方式，不要使用大小写与下划线混排的方式
	可读性
1	4-1：注意运算符的优先级，并用括号明确表达式的操作顺序，避免使用默认优先级
2	4-2：避免使用不易理解的数字，用有意义的标识来替代。涉及物理状态或者含有物理意义的常量，不应直接使用数字，必须用有意义的枚举或宏来代替
	变量
1	5-1：去掉没必要的公共变量
2	5-2：仔细定义并明确公共变量的含义、作用、取值范围及公共变量间的关系
3	5-3：明确公共变量与操作此公共变量的函数或过程的关系，如访问、修改及创建等
4	5-4：当向公共变量传递数据时，要十分小心，防止赋予不合理的值或越界等现象发生
5	5-5：防止局部变量与公共变量同名
6	5-6：严禁使用未经初始化的变量作为右值
	函数、过程
1	6-1：对所调用函数的错误返回码要仔细、全面地处理
2	6-2：明确函数功能，精确（而不是近似）地实现函数设计
3	6-3：编写可重入函数时，应注意局部变量的使用（如编写 C/C++语言的可重入函数时，应使用 auto 即缺省态局部变量或寄存器变量）

续表

序　号	总 则 条 款
	函数、过程
4	6-4：编写可重入函数时，若使用全局变量，则应通过关中断、信号量（即 P、V 操作）等手段对其加以保护
	可测性
1	7-1：在同一项目组或产品组内，要有一套统一的为集成测试与系统联调准备的调测开关及相应打印函数，并且要有详细的说明
2	7-2：在同一项目组或产品组内，调测打印出的信息串的格式要有统一的形式。信息串中至少要有所在模块名（或源文件名）及行号
3	7-3：编程的同时要为单元测试选择恰当的测试点，并仔细构造测试代码、测试用例，同时给出明确的注释说明。测试代码部分应作为（模块中的）一个子模块，以方便测试代码在模块中的安装与拆卸（通过调测开关）
4	7-4：在进行集成测试/系统联调之前，要构造好测试环境、测试项目及测试用例，同时仔细分析并优化测试用例，以提高测试效率
5	7-5：使用断言来发现软件问题，提高代码可测性
6	7-6：用断言来检查程序正常运行时不应发生，但在调测时有可能发生的非法情况
7	7-7：不能用断言来检查最终产品肯定会出现且必须处理的错误情况
8	7-8：为较复杂的断言加上明确的注释
9	7-9：用断言确认函数的参数
10	7-10：用断言保证没有定义的特性或功能不被使用
11	7-11：用断言对程序开发环境（OS/Compiler/Hardware）的假设进行检查
12	7-12：正式软件产品中应把断言及其他调测代码去掉（即把有关的调测开关关掉）
13	7-13：在软件系统中设置与取消有关测试手段，不能对软件实现的功能等产生影响
14	7-14：用调测开关来切换软件的 DEBUG 版和正式版，而不要同时存在正式版本和 DEBUG 版本的不同源文件，以减少维护的难度
15	7-15：软件的 DEBUG 版本和发行版本应该统一维护，不允许分家，并且要时刻注意保证两个版本在实现功能上的一致性
	程序效率
1	8-1：编程时要经常注意代码的效率
2	8-2：在保证软件系统的正确性、稳定性、可读性及可测性的前提下，提高代码效率
3	8-3：局部效率应为全局效率服务，不能因为提高局部效率而对全局效率造成影响
4	8-4：通过对系统数据结构的划分与组织的改进，以及对程序算法的优化来提高空间效率
5	8-5：循环体内工作量最小化
	质量保证
1	9-1：在软件设计过程中构筑软件质量
2	9-2：代码质量保证优先原则
3	9-3：只引用属于自己的存储空间
4	9-4：防止引用已经释放的内存空间



续表

序　号	总 则 条 款
	质量保证
5	9-5：过程/函数中分配的内存，在过程/函数退出之前要释放
6	9-6：过程/函数中申请的（为打开文件而使用的）文件句柄，在过程/函数退出之前要关闭
7	9-7：防止内存操作越界
8	9-8：认真处理程序所能遇到的各种出错情况
9	9-9：系统运行之初，要初始化有关变量及运行环境，防止未经初始化的变量被引用
10	9-10：系统运行之初，要对加载到系统中的数据进行一致性检查
11	9-11：严禁随意更改其他模块或系统的有关设置和配置
12	9-12：不能随意改变与其他模块的接口
13	9-13：充分了解系统的接口之后，再使用系统提供的功能
14	9-14：编程时，要防止差 1 错误
15	9-15：要时刻注意易混淆的操作符。当编完程序后，应从头至尾检查一遍这些操作符，以防止拼写错误
16	9-16：如果有可能，if 语句尽量加上 else 分支，对没有 else 分支的语句要小心对待；switch 语句必须有 default 分支
	代码编辑、编译、审查
1	10-1：打开编译器的所有告警开关对程序进行编译
2	10-2：在产品软件（项目组）中，要统一编译开关选项
3	10-3：通过代码走读及审查方式对代码进行检查
4	10-4：测试部测试产品之前，应对代码进行抽查及审查
	代码测试、维护
1	11-1：单元测试要求至少达到语句覆盖
2	11-2：单元测试开始要跟踪每一条语句，并观察数据流及变量的变化
3	11-3：清理、整理或优化后的代码要经过审查及测试
4	11-4：代码版本升级要经过严格测试
5	11-5：使用工具软件对代码版本进行维护
6	11-6：正式版本上软件的任何修改都应有详细的文档记录
	宏
1	12-1：用宏定义表达式时，要使用完备的括号
2	12-2：将宏所定义的多条表达式放在大括号中
3	12-3：使用宏时，不允许参数发生变化

4. 代码检查规则

在代码检查中，需要依据被测软件的特点，选用适当的标准与规则规范。在使用测试软件进行自动化代码检查时，测试工具一般会内置许多的编码规则，例如，Parasoft 公司用于 C/C++语言测试的 C++Test，内置了 5 类规则：通用规则、C++编码规则、C 编

码规则、Meyers-Klaus 规则和自定义规则。我们需要根据编程语言以及被测程序的特点，挑选适当的规则进行检查。

例如，在测试中，可以选择表 6-2 中的规则进行自动化测试，在此基础上使用桌面检查、代码走查、代码审查等人工检查的方法仔细检查程序的结构、逻辑等方面的缺陷。

表 6-2　代码检查规则

序　号	规则描述	重要程度
1	if,else,while,do 语句后面必须跟一个语句块，哪怕是空的	一般
2	使用 float 或 double 类型变量比较时，用“<=”和“=>”代替“==”	重要
3	初始化所有变量	一般
4	不要使用 goto 语句	一般
5	检查 new 的返回值	重要
6	在混合表达式中，所有的运算符应该应用于相同的数据类型之间	重要
7	不要比较不同类型的指针变量	重要
8	初始化所有的指针变量	一般
9	代码行的长度小于 80 个字符	一般
10	成员函数中避免使用全局变量	一般
11	限制函数调用的数目	一般
12	限制每个函数中代码块的数目	一般
13	避免使用全局变量	一般
14	为防止编译器生成一个构造函数，至少声明一个构造函数	一般
15	不要把常量转变为非常量	一般

5．缺陷检查表

在进行人工代码检查时，代码缺陷检查表是我们用到的测试用例。

代码缺陷检查表中一般包括容易出错的地方和在以往的工作中遇到的典型错误，如下所示。

- 格式部分：

① 嵌套的 IF 正确地缩进了吗？

② 注释准确并有意义吗？

③ 使用有意义的标号了吗？

④ 代码基本上与开始时的模块模式一致吗？

⑤ 遵循全套的编程标准吗？

- 入口和出口的连接：

① 初始入口和最终出口正确吗？

② 当对另一个模块的每一次调用时，全部所需的参数传送给每一个被调用的模块吗？

③ 被传送的参数值正确地设置了吗？

④ 对关键的被调用模块的意外情况（如丢失、混乱）有处理吗？

- 程序语言的使用：

① 使用一个或一组最佳的动词了吗？

② 模块中使用完整定义的语言的有限子集了吗？

③ 使用了适当的跳转语句吗？

- 存储器使用：

① 每一个域在第一次使用前正确地初始化了吗？

② 规定的域正确吗？

③ 每个域有正确的变量类型声明吗？

- 判断和转移：

① 判断正确的条件了吗？

② 用于判断的是正确的变量吗？

③ 每个转移目标正确并至少执行一次了吗？

- 性能：

① 逻辑被最佳地编码了吗？

② 提供正式的错误/例外子程序了吗？

- 可维护性：

① 清单格式适于提高可读性吗？

② 标号和子程序符合代码的逻辑意义吗？

- 逻辑：

① 全部设计已实现了吗？

② 代码做的是设计规定的内容吗？

③ 每一个循环执行正确的次数了吗？

- 可靠性：

① 对从外部接口采集的数据有确认吗？

② 遵循可靠性编程要求了吗？

对应于不同的编程语言，代码缺陷检查表的具体内容将会不同。例如，针对广泛使用的 C/C++语言，可以参照表 6-3 所示的代码缺陷检查表。

表 6-3　代码缺陷检查表

	重要性	审查项	结论
文件结构		头文件和定义文件的名称是否合理	
		头文件和定义文件的目录结构是否合理	
		版权和版本声明是否完整	
	重要	头文件是否使用了 ifndef/define/endif 预处理块	
		头文件中是否只存放“声明”而不存放“定义”	
		……	
程序的版式		空行是否得体	
		代码行内的空格是否得体	
		长行拆分是否得体	
		“{”和“}”是否各占一行并且对齐于同一列	
	重要	一行代码是否只做一件事？如只定义一个变量，只写一条语句	
	重要	if、for、while、do 等语句自占一行，不论执行语句多少都要加“{}”	
	重要	在定义变量（或参数）时，是否将修饰符 * 和 & 紧靠变量名	
		注释是否清晰并且必要	
	重要	注释是否有错误或者可能导致误解	
	重要	类结构的 public, protected, private 顺序是否在所有的程序中保持一致	
		……	
命名规则	重要	命名规则是否与所采用的操作系统或开发工具的风格保持一致	
		标识符是否直观且可以拼读	
		标识符的长度应当符合“min-length && max-information”原则	
	重要	程序中是否出现相同的局部变量和全部变量	
		类名、函数名、变量和参数、常量的书写格式是否遵循一定的规则	
		静态变量、全局变量、类的成员变量是否加前缀	
		……	
表达式与基本语句	重要	如果代码行中的运算符比较多，是否已经用括号清楚地确定表达式的操作顺序	
		是否编写太复杂或者多用途的复合表达式	
	重要	是否将复合表达式与“真正的数学表达式”混淆	
	重要	是否用隐含错误的方式写 if 语句？例如 （1）将布尔变量直接与 TRUE、FALSE 或者 1、0 进行比较 （2）将浮点变量用“==”或“！=”与任何数字比较 （3）将指针变量用“==”或“！=”与 NULL 比较	
		如果循环体内存在逻辑判断，并且循环次数很大，是否已经将逻辑判断移到循环体的外面	



续表

	重要性	审查项	结论
表达式与基本语句	重要	case 语句的结尾是否忘了加 break	
	重要	是否忘记写 switch 的 default 分支	
	重要	使用 goto 语句时是否留下隐患？例如跳过了某些对象的构造、变量的初始化、重要的计算等	
		……	
常量		是否使用含义直观的常量来表示那些将在程序中多次出现的数字或字符串	
		在 C++ 程序中，是否用 const 常量取代宏常量	
	重要	如果某一常量与其他常量密切相关，是否在定义中包含了这种关系	
		是否误解了类中的 const 数据成员？因为 const 数据成员只在某个对象生存期内是常量，而对于整个类而言却是可变的	
		……	
函数设计		参数的书写是否完整？不要贪图省事只写参数的类型而省略参数名字	
		参数命名、顺序是否合理	
		参数的个数是否太多	
		是否使用类型和数目不确定的参数	
		是否省略了函数返回值的类型	
		函数名字与返回值类型在语义上是否冲突	
	重要	是否将正常值和错误标志混在一起返回？正常值应当用输出参数获得，而错误标志用 return 语句返回	
	重要	在函数体的“入口处”，是否用 assert 对参数的有效性进行检查	
	重要	是否滥用了 assert？例如混淆非法情况与错误情况，后者是必然存在的并且是一定要作出处理的	
	重要	return 语句是否返回指向“栈内存”的“指针”或者“引用”	
		是否使用 const 提高函数的健壮性？const 可以强制保护函数的参数、返回值，甚至函数的定义体。“Use const whenever you need”	
		……	
内存管理	重要	用 malloc 或 new 申请内存之后，是否立即检查指针值是否为 NULL（防止使用指针值为 NULL 的内存）	
	重要	是否忘记为数组和动态内存赋初值（防止将未被初始化的内存作为右值使用）	
	重要	数组或指针的下标是否越界	
	重要	动态内存的申请与释放是否配对（防止内存泄漏）	
	重要	是否有效地处理了“内存耗尽”问题	
	重要	是否修改“指向常量的指针”的内容	

续表

	重要性	审查项	结论
内存管理	重要	是否出现野指针，例如 （1）指针变量没有被初始化 （2）用 free 或 delete 释放了内存之后，忘记将指针设置为 NULL	
	重要	是否将 malloc/free 和 new/delete 混淆使用	
	重要	malloc 语句是否正确无误？例如字节数是否正确？类型转换是否正确	
	重要	在创建与释放动态对象数组时，new/delete 的语句是否正确无误	
		……	
C++ 函数的高级特性		重载函数是否有二义性	
	重要	是否混淆了成员函数的重载、覆盖与隐藏	
		运算符的重载是否符合制定的编程规范	
		是否滥用了内联函数？例如函数体内的代码比较长，函数体内出现循环	
	重要	是否用内联函数取代了宏代码	
		……	
类的构造函数、析构函数和赋值函数	重要	是否违背编程规范而让 C++ 编译器自动为类产生四个缺省的函数：（1）缺省的无参数构造函数；（2）缺省的拷贝构造函数；（3）缺省的析构函数；（4）缺省的赋值函数	
	重要	构造函数中是否遗漏了某些初始化工作	
	重要	是否正确地使用了构造函数的初始化表	
	重要	析构函数中是否遗漏了某些消除工作	
		是否错写、错用了拷贝构造函数和赋值函数	
	重要	赋值函数一般分四个步骤：（1）检查自赋值；（2）释放原有内存资源；（3）分配新的内存资源，并复制内容；（4）返回 *this。是否遗漏了重要步骤	
	重要	是否正确地编写了派生类的构造函数、析构函数、赋值函数？注意事项： （1）派生类不可能继承基类的构造函数、析构函数、赋值函数 （2）派生类的构造函数应在其初始化表里调用基类的构造函数 （3）基类与派生类的析构函数应该为虚（即加 virtual 关键字） （4）在编写派生类的赋值函数时，注意不要忘记对基类的数据成员重新赋值	
		……	
类的高级特性	重要	是否违背了继承和组合的规则 （1）若在逻辑上 B 是 A 的“一种”，并且 A 的所有功能和属性对 B 而言都有意义，则允许 B 继承 A 的功能和属性 （2）若在逻辑上 A 是 B 的“一部分”（a part of），则不允许 B 从 A 派生，而是要用 A 和其他东西组合出 B	
		……	



续表

	重要性	审查项	结论
其他常见问题	重要	数据类型问题： （1）变量的数据类型有错误吗 （2）存在不同数据类型的赋值吗 （3）存在不同数据类型的比较吗	
	重要	变量值问题： （1）变量的初始化或缺省值有错误吗 （2）变量发生上溢或下溢吗 （3）变量的精度够吗	
	重要	逻辑判断问题： （1）由于精度原因导致比较无效吗 （2）表达式中的优先级有误吗 （3）逻辑判断结果颠倒吗	
	重要	循环问题： （1）循环终止条件不正确吗 （2）无法正常终止（死循环）吗 （3）错误地修改循环变量吗 （4）存在误差累积吗	
	重要	错误处理问题： （1）忘记了进行错误处理吗 （2）是否错误处理程序块一直没有机会被运行 （3）错误处理程序块本身就有毛病吗？如报告的错误与实际错误不一致，处理方式不正确等 （4）错误处理程序块是“马后炮”吗？如在被它被调用之前软件已经出错	
	重要	文件 I/O 问题： （1）对不存在的或者错误的文件进行操作吗 （2）文件以不正确的方式打开了吗 （3）文件结束判断不正确吗 （4）没有正确地关闭文件吗	

6.2.2 静态结构分析法

程序的结构形式是白盒测试的主要依据。研究表明程序员 38%的时间花费在理解软

件系统上，因为代码以文本格式被写入多重文件中，这是很难阅读理解的，需要其他一些东西来帮助人们阅读理解，如各种图表等，而静态结构分析满足了这样的需求。

在静态结构分析中，测试者通过使用测试工具分析程序源代码的系统结构、数据结构、数据接口、内部控制逻辑等内部结构，生成函数调用关系图、模块控制流图、内部文件调用关系图、子程序表、宏和函数参数表等各类图形图表，可以清晰地标识整个软件系统的组成结构，使其便于阅读与理解，然后可以通过分析这些图表，检查软件有没有存在缺陷或错误。

其中函数调用关系图通过应用程序中各函数之间的调用关系展示了系统的结构。通过查看函数调用关系图，可以检查函数之间的调用关系是否符合要求，是否存在递归调用，函数的调用层次是否过深，有没有存在孤立的没有被调用的函数。从而可以发现系统是否存在结构缺陷，发现哪些函数是重要的，哪些是次要的，需要使用什么级别的覆盖要求……

模块控制流图是与程序流程图相类似的由许多节点和连接结点的边组成的一种图形，其中一个节点代表一条语句或数条语句（图中的各种图形符号代表不同的意义），边表示节点间的控制流向，它显示了一个函数的内部逻辑结构。如图 6-5 所示是测试工具 Logiscope 所使用的基本图例，如图 6-6（a）、（b）、（c）所示是典型逻辑结构所对应的控制流图基本结构。

模块控制流图可以直观地反映出一个函数的内部逻辑结构，通过检查这些模块控制流图，能够很快发现软件的错误与缺陷。

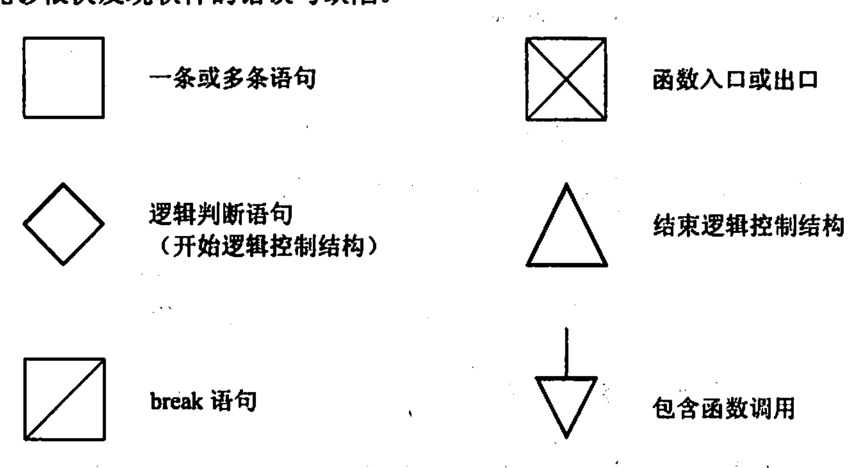


图 6-5　Logiscope 基本图例

while-do-end of
repeat until
infinite
ITERATIVE STRUCTURES
(a)

if then-end if
if then-else-end if
case of-end of case
switch(in C)
if then-elsif-end if
SELECTIVE STRUCTURES
(b)

branch
conditional branch
main entry/auxiliary entry
auxiliary exit main exit
SPECIFI C STRUCTURES
(c)

图 6-6 控制流图基本结构

例如：如图 6-7 所示是某 GIS 软件中某个函数的模块控制流图。

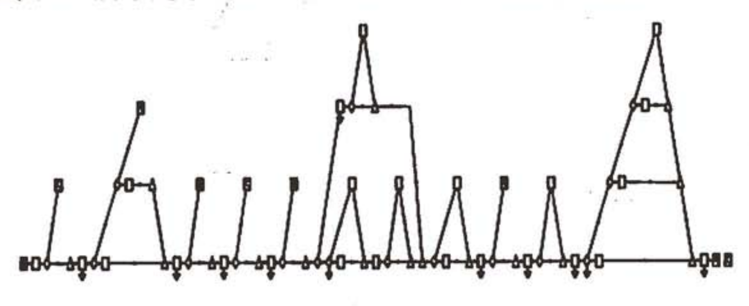


图 6-7 GIS 模块控制流图

该函数的结构存在重大缺陷：首先，我们可以看到在该函数中存在无法执行的死代码，即图中最右边的孤立出口。其次，该函数有多达 8 个出口，其中一个属于无法到达的出口。由于可能没有在所有的出口进行动态内存的释放与回收操作，因此这样的结构存在内存泄漏的可能。

如图 6-8 所示是某 MIS 软件中某个函数的模块控制流图。

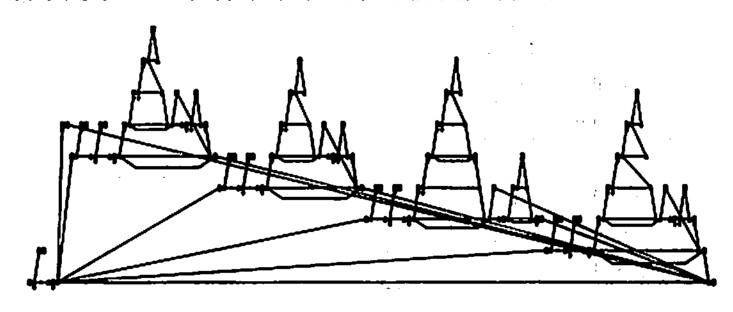


图 6-8　MIS 模块控制流图

该函数的结构也同样存在重大缺陷：

首先，该函数有多个出口，存在内存泄漏的可能。其次，该函数结构复杂，有超过 10 个逻辑判断结点，在这些结点上出现逻辑错误的概率将大大增加，将降低其可靠性，而且过多的逻辑判断结点可能会破坏对 CPU 操作进行优化的处理，影响其运行性能。

6.2.3　静态质量度量法

根据 ISO/IEC 9126 国际标准的定义，软件的质量包括以下六个方面：

- 功能性（FUNCTIONALITY）；
- 可靠性（RELIABILITY）；
- 可用性（USABILITY）；
- 有效性（EFFICIENCY）；
- 可维护性（MAINTAINABILITY）；
- 轻便性（PORTABILITY）。

以 ISO 9126 质量模型作为基础，我们可以构造质量度量模型，用于评估软件的每个方面。例如，按以下方法构造的质量模型可以度量程序的可维护性（maintainability）。首先，该模型从上到下分为 3 层：质量因素（factors）、分类标准（criteria）和度量规则（metrics）。其中质量因素对应 ISO 9126 质量模型的质量特性，分类标准对应 ISO 9126 质量模型的子特性，度量规则用于规范软件的各种行为属性。其次，按以下方式定义各参数及计算公式。



- 度量规则（Metrics）。

度量规则使用了代码行数、注释频度等参数度量软件的各种行为属性，具体参数定义如表 6-4 所示。

表 6-4 度量规则参数表

简 称	名 称	最小值	最大值
PARA	Number of function parameters	0	5
PATH	Number of paths	1	80
LEVL	Number of levels	0	4
DRCT_CALLS	Number of direct calls	0	7
RETU	Number of RETURN statements	0	1
NBCALLING	Number of callers	0	5
GOTO	Number of GOTO statements	0	0
VOCF	Vocabulary frequency	1	4
LVAR	Number of local variables	0	5
COMF	Comments frequency	0.2	∞
AVGS	Average size of statements	1	9
STMT	Number of statements	1	50
VG	Cyclomatic number (VG)	1	10
IND_CALLS	Number of relative call graph call-paths	1	30
TESTBTY	Relative call graph System testability	0	1
HIER_CPX	Relative call graph Hierarchical complexity	1	5
LEVELS	Number of relative call graph levels	1	12
STRU_CPX	Relative call graph Structural complexity	0	3
DDP	DC coverage rate	0	100
IB	SC coverage rate	0	100
PPP	Procedure-to-Procedure Path coverage rate	0	100

- 分类标准（criteria）。

软件的可维护性采用以下四个分类标准来评估：

① 可分析性（ANALYZABILITY）

② 可修改性（CHANGEABILITY）

③ 稳定性（STABILITY）

④ 可测性（TESTABILITY）。

每个分类标准由一系列度量规则组成，各个规则分配一个权重，由规则的取值与权重值计算出每个分类标准的取值。各分类标准组成如表 6-5 所示。

表 6-5　分类标准组

分类标准	度量规则	权重
function_TESTABILITY	PARA	1
	PATH	1
	LEVL	1
	DRCT_CALLS	1
function_STABILITY	PARA	1
	DRCT_CALLS	1
	RETU	1
	NBCALLING	1
function_CHANGEABILITY	GOTO	1
	VOCF	1
	LVAR	1
	PARA	1
function_ANALYZABILITY	COMF	1
	AVGS	1
	STMT	1
	VG	1
relativeCall_ANALYZABILITY	LEVELS	1
	STRU_CPX	1
relativeCall_STABILITY	HIER_CPX	1
	IND_CALLS	1
relativeCall_TESTABILITY	IND_CALLS	1
	TESTBTY	1

各分类标准的结果按以下标准区分等级，如表 6-6 至表 6-12 所示。

function_TESTABILITY = DRCT_CALLS + LEVL + PATH + PARA

表 6-6　function_TESTABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	4	4	FAIR	2	2
GOOD	3	3	POOR	0	1

function_STABILITY = NBCALLING + RETU + DRCT_CALLS + PARA



表 6-7 function_STABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	4	4	FAIR	2	2
GOOD	3	3	POOR	0	1

function_CHANGEABILITY = PARA + LVAR + VOCF + GOTO

表 6-8 function_CHANGEABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	4	4	FAIR	2	2
GOOD	3	3	POOR	0	1

function_ANALYZABILITY = VG + STMT + AVGS + COMF

表 6-9 function_ANALYZABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	4	4	FAIR	2	2
GOOD	3	3	POOR	0	1

relativeCall_ANALYZABILITY = STRU_CPX + LEVELS

表 6-10 relativeCall_ANALYZABILITY 的等级划分

等级描述	上限	下限
EXCELLENT	2	2
GOOD	1	1
POOR	0	0

relativeCall_STABILITY = CALL_PATHS + HIER_CPX

表 6-11 relativeCall_STABILITY 的等级划分

等级描述	上限	下限
EXCELLENT	2	2
GOOD	1	1
POOR	0	0

relativeCall_TESTABILITY = TESTBTY + CALL_PATHS

表 6-12　relativeCall_TESTABILITY 的等级划分

等级描述	上限	下限
EXCELLENT	2	2
GOOD	1	1
POOR	0	0

这样，依据这些标准和最终测试结果，可将代码的质量分成四个等级。

① 优秀（EXCELLENT）：符合本模型框架中的所有规则。

② 良好（GOOD）：未大量偏离模型框架中的规则。

③ 一般（FAIR）：违背了模型框架中的大量规则。

④ 较差（POOR）：无法保障正常的软件可维护性。

其中前三者被认为是可以接受的，最后一个等级则是不可接受的。

- 质量因素（factors）。

质量因素的取值与分类标准的计算方式相似：依据各分类标准取值组合权重方法来计算，如表 6-13 所示。

表 6-13　质量因素权重计算表

质 量 因 素	分 类 标 准	权 重
function_MAINTAINABILITY	function_TESTABILITY	1
	function_STABILITY	1
	function_CHANGEABILITY	1
	function_ANALYZABILITY	1
relativeCall_MAINTAINABILITY	relativeCall_TESTABILITY	1
	relativeCall_STABILITY	1
	relativeCall_ANALYZABILITY	1

同样，依据质量因素取值，也将其分成四个等级：优秀（EXCELLENT）、良好（GOOD）、一般（FAIR）和较差（POOR），其中前三者被认为是可以接受的，最后一个等级则是不可接受的。

如表 6-14 和表 6-15 所示为 function_MAINTAINABILITY 和 relative Call_MINTAINABILITY 的等级划分。

```
function_MAINTAINABILITY = function_ANALYZABILITY
                         + function_CHANGEABILITY
                         + function_STABILITY
                         + function_TESTABILITY
```

表 6-14 function_MAITAINABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	12	12	FAIR	4	7
GOOD	8	11	POOR	0	3

relativeCall_MAINTAINABILITY = relativeCall_ANALYZABILITY
+ relativeCall_STABILITY
+ relativeCall_TESTABILITY

表 6-15 relativeCall_MAINTAINABILITY 的等级划分

等级描述	上限	下限	等级描述	上限	下限
EXCELLENT	6	6	FAIR	1	3
GOOD	4	5	POOR	0	0

将上述质量模型应用于被测程序后，就可以通过量化的数据对软件的质量进行评估了。

6.2.4 逻辑覆盖法

白盒测试的动态测试要根据程序的控制结构设计测试用例，其原则是：

- 保证一个模块中的所有独立路径至少被使用一次；
- 对所有逻辑值均需测试 true 和 false；
- 在上下边界及可操作范围内运行所有循环；
- 检查内部数据结构以确保其有效性。

但是对一个具有多重选择和循环嵌套的程序，不同的路径数目可能是天文数字。而且即使精确地实现了白盒测试，也不能断言测试过的程序完全正确。如图 6-9 所示的穷举测试流程图，其中包括了一个执行达 20 次的循环，它所包含的不同执行路径数高达 5^{20} 条，假使有这么一个测试程序，对每一条路径进行测试需要 1ms，假设一天工作 24 小时，一年工作 365 天，若要对它进行穷举测试，也需要 3024 年的时间。

以上的情况说明，实现穷举测试的工作量过大，需要的时间过长，实施起来是不现实的。任何软件开发项目都要受到期限、费用、人力和机时等条件的限制，尽管我们以为为了充分揭露程序中的所有隐藏错误，彻底的做法是针对所有可能的数据进行测试，但事实告诉人们，这样做是不可能的。

在测试阶段既然穷举测试不可行，为了节省时间和资源，提高测试效率，就必须精心设计测试用例，也就是从数量巨大的可用测试用例中精心挑选少量的测试数据，使得采用这些测试数据就能够达到最佳的测试效果。

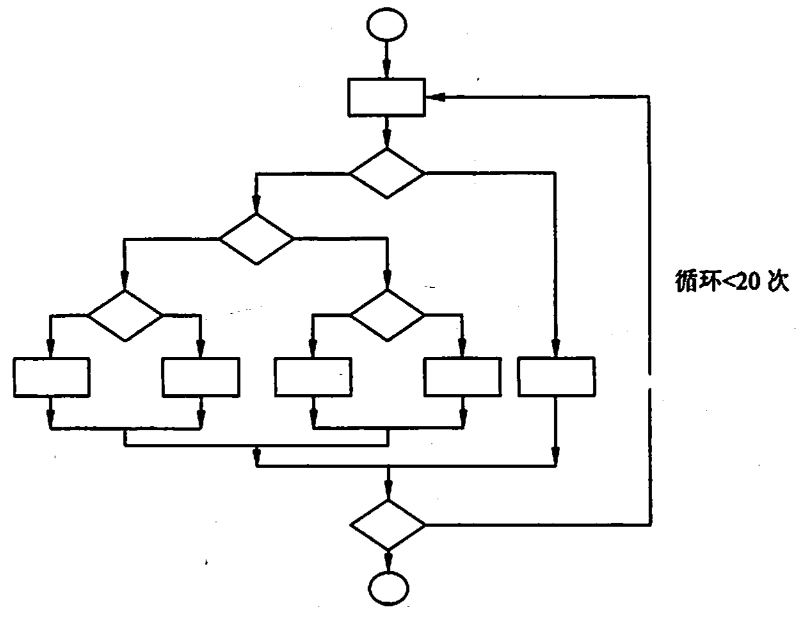


图 6-9　穷举测试

本节和下节将介绍几种实用的白盒测试用例设计方法：逻辑覆盖法和基本路径测试法。

逻辑覆盖是通过对程序逻辑结构的遍历实现程序的覆盖。它是一系列测试过程的总称，这组测试过程逐渐进行越来越完整的通路测试。从覆盖源程序语句的详尽程度分析，逻辑覆盖标准包括以下不同的覆盖标准：语句覆盖（SC）、判定覆盖（DC）、条件覆盖（CC）、条件判定组合覆盖（CDC）、多条件覆盖（MCC）和修正判定条件覆盖（MCDC）。

为便于理解，我们使用如下所示的程序（用 C 语言书写），如图 6-10 所示的是其流程图。

[程序]：

```
int function1(bool a, bool b, bool c)
{
  int x;
  x = 0;
  if ( a && ( b || c ) )
      x = 1;
  return x;
}
```



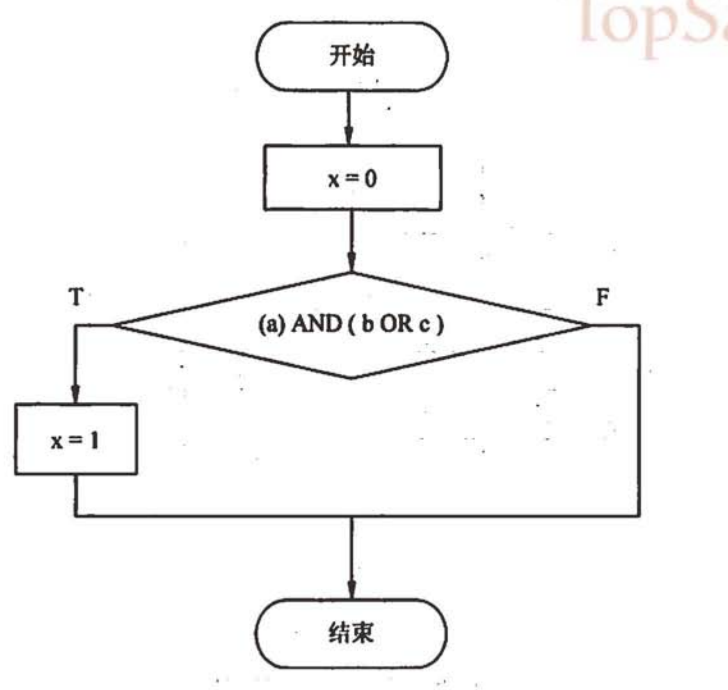


图 6-10 参考例子流程图

1. 语句覆盖（SC）

为了暴露程序中的错误，程序中的每条语句至少应该执行一次。因此，语句覆盖（Statement Coverage）的含义是：选择足够多的测试数据，使被测程序中每条语句至少执行一次。

为了使上述程序中的每条语句都能够至少执行一次，我们可以构造以下测试用例即可实现：

a = T，b = T，c = T。

从程序中的每条语句都得到执行这一点看，语句覆盖的方法似乎能够比较全面地检验每一条语句，但是语句覆盖对程序执行逻辑的覆盖很低，这是其最严重的缺陷。

假如，这一程序段中判定的逻辑运算有问题，例如，判定的第一个运算符“&&”错写成运算符“||”，或第二个运算符“||”错写成运算符“&&”，这时使用上述的测试用例仍然可以达到100%的语句覆盖，上述的逻辑错误无法发现。

因此一般认为语句覆盖是很弱的逻辑覆盖。

2. 判定覆盖（DC）

比语句覆盖稍强的覆盖标准是判定覆盖（Decision Coverage）。判定覆盖的含义是：

设计足够的测试用例，使得程序中的每个判定至少都获得一次“真值”或“假值”，或者说使得程序中的每一个取“真”分支和取“假”分支至少经历一次，因此判定覆盖又称为分支覆盖。

除了双值的判定语句外，还有多值判定语句，如case语句。因此判定覆盖更一般的含义是：使得每一个判定获得每一种可能的结果至少一次。

以上述代码为例，构造以下测试用例即可实现判定覆盖标准：

- a＝T，b＝T，c＝T。
- a＝F，b＝F，c＝F。

应该注意到，上述两组测试用例不仅满足了判定覆盖，而且满足了语句覆盖，从这一点看，判定覆盖要比语句覆盖更强一些。但是同样地，假如这一程序段中判定的逻辑运算有问题，如表6-16所示，判定的第一个运算符“&&”错写成运算符“||”或第二个运算符“||”错写成运算符“&&”，这时使用上述的测试用例可以达到100%的判定覆盖，仍然无法发现上述的逻辑错误。因此需要更强的逻辑覆盖标准。

表6-16　判定覆盖

序号	a	b	c	a && (b || c)	a || (b || c)	判定覆盖/%
1	T	T	T	T	T	50
2	F	F	F	F	F	50

3．条件覆盖（CC）

在设计程序中，一个判定语句是由多个条件组合而成的复合判定，在如图6-10所示参考例子流程图的程序中，判定（a）AND（b OR c）包含了三个条件：a，b和c。为了更彻底地实现逻辑覆盖，可以采用条件覆盖（Condition Coverage）的标准。条件覆盖的含义是：构造一组测试用例，使得每一判定语句中每个逻辑条件的可能值至少满足一次。

按照这一定义，上述例子要达到100%的条件覆盖，可以使用以下测试用例：

- a＝F，b＝T，c＝F。
- a＝T，b＝F，c＝T。

仔细分析可以发现，上述用例在满足条件覆盖的同时，把判定的两个分支也覆盖了，这样是否可以说，达到了条件覆盖也就必然实现了判定覆盖呢？

假如选用以下的两组测试用例：

- a＝F，b＝T，c＝T。
- a＝T，b＝F，c＝F。

我们会发现覆盖了条件的测试用例并没有覆盖分支，如表6-17所示。为解决这一矛盾，需要对条件和分支兼顾。



表 6-17 条件覆盖

序号	a	b	c	a && (b ‖ c)	条件覆盖/%	判定覆盖/%
1	F	T	T	F	100	50
2	T	F	F	F		

4. 条件判定组合覆盖（CDC）

条件判定组合覆盖的含义是：设计足够的测试用例，使得判定中每个条件的所有可能（真/假）至少出现一次，并且每个判定本身的判定结果（真/假）也至少出现一次。

对于如图 6-10 所示的例子，选用以下的两组测试用例可以符合条件判定组合覆盖标准：

- a = T，b = T，c = T。
- a = F，b = F，c = F。

但是条件判定组合覆盖也存在一定的缺陷，例如，判定的第一个运算符“&&”错写成运算符“‖”或第二个运算符“‖”错写成运算符“&&”，如表 6-18 所示，这时使用上述的测试用例仍然可以达到 100%的条件判定组合覆盖，上述的逻辑错误无法发现。

表 6-18 条件判定组合覆盖

序号	a	b	c	a && (b ‖ c)	a && (b && c)	条件判定组合覆盖/%
1	T	T	T	T	T	100
2	F	F	F	F	F	

5. 多条件覆盖（MCC）

多条件覆盖也称条件组合覆盖，它的含义是：设计足够的测试用例，使得每个判定中条件的各种可能组合都至少出现一次。显然满足多条件覆盖的测试用例是一定满足判定覆盖、条件覆盖和条件判定组合覆盖的。

对于如图 6-19 所示的例子，判定语句中有三个逻辑条件，每个逻辑条件有两种可能取值，因此共有 $2^3=8$ 种可能组合，如表 6-19 所示的测试用例保证了多条件覆盖。

表 6-19 多条件覆盖

序号	a	b	c	a && (b ‖ c)
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

由上可知，当一个程序中判定语句较多时，其条件取值的组合数目是非常大的。

6. 修正条件判定覆盖（MCDC）

修正条件判定覆盖是由欧美的航空/航天制造厂商和使用单位联合制定的“航空运输和装备系统软件认证标准”，目前在国外的国防、航空航天领域应用广泛。这个覆盖度量需要足够的测试用例来确定各个条件能够影响到包含的判定的结果。它要求满足两个条件：首先，每一个程序模块的入口和出口点都要考虑至少要被调用一次，每个程序的判定到所有可能的结果值要至少转换一次；其次，程序的判定被分解为通过逻辑操作符（and、or）连接的 bool 条件，每个条件对于判定的结果值是独立的。

对于如图 6-10 所示的例子，可以设计如表 6-20 中的 8 个用例，在此基础上，按照 MCDC 的要求选择需要的用例。

表 6-20　修正条件判定覆盖

序号	a	b	c	a && (b || c)	a	b	c
1	T	T	T	T	5		
2	T	T	F	T	6	4	
3	T	F	T	T	7		4
4	T	F	F	F		2	3
5	F	T	T	F	1		
6	F	T	F	F	2		
7	F	F	T	F	3		
8	F	F	F	F			

从表中我们可以看出，布尔变量 a 可以通过用例 1 和 5 达到 MCDC 的要求（用例 2 和 6 或用例 3 和 7 也可以满足相应要求），变量 b 可以通过用例 2 和 4 达到 MCDC 的要求，变量 c 可以通过用例 3 和 4 达到 MCDC 的要求，因此使用用例集{1，2，3，4，5}即可满足 MCDC 的要求。显而易见，这不是惟一的用例组合。

6.2.5　基本路径测试法

上节的例子是个比较简单的程序段，只有两条路径。但在实际问题中，一个不太复杂的程序，其路径的组合都是一个庞大的数字。如图 6-9 所示的穷举测试程序竟有 5^{20} 条路径，要在测试中覆盖这样多的路径是不现实的。

为解决这一难题，需要把覆盖的路径数压缩到一定限度内，例如，程序中的循环体只执行一次。本节介绍的基本路径测试就是这样一种测试方法，它在程序控制流图的基础上，通过分析控制流图的环路复杂性，导出基本可执行路径的集合，然后据此设计测试用例。设计出的测试用例要保证在测试中程序的每一条可执行语句至少执行一次。



1．程序的控制流图

控制流图是描述程序控制流的一种图示方式。其中基本的控制结构对应的图形符号如图 6-11 所示。在如图 6-11 所示的图形符号中，圆圈称为控制流图的一个结点，它表示一个或多个无分支的语句或源程序语句。

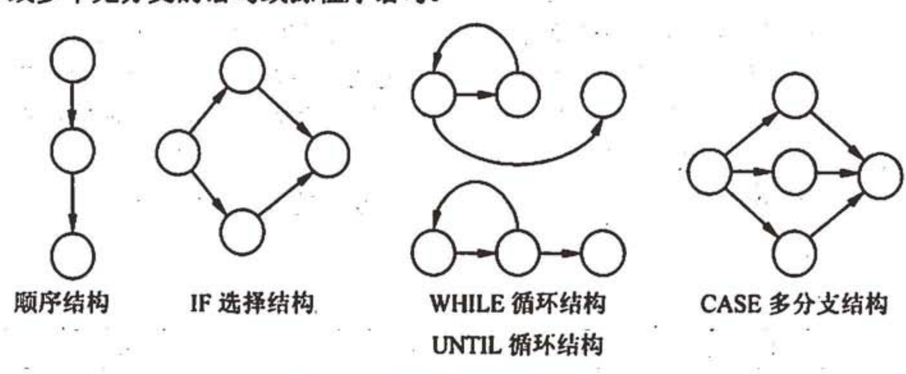


图 6-11 控制流程图的图形符号

如图 6-12（a）所示的是一个程序的流程图，它可以映射成如图 6-12（b）所示的控制流程图。

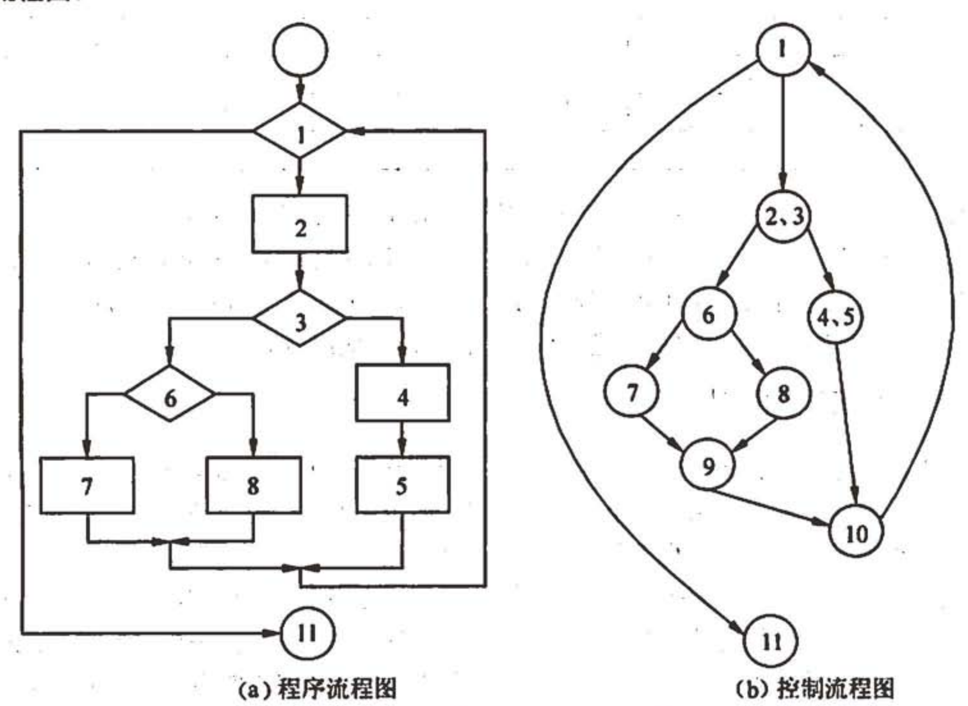


图 6-12 程序流程图和对应的控制流程图

这里我们假定在流程图中用菱形框表示的判定条件内没有复合条件，而一组顺序处理框可以映射为一个单一的结点。控制流程图中的箭头（边）表示了控制流的方向，类似于流程图中的流线，一条边必须终止于一个结点，但在选择或者是多分支结构中分支的汇聚处，即使汇聚处没有执行语句也应该添加一个汇聚结点。边和结点圈定的部分叫区域，当对区域计数时，图形外的部分也应记为一个区域。

如果判断中的条件表达式是复合条件，即条件表达式是由一个或多个逻辑运算符（or、and、nand 和 nor）连接的逻辑表达式，则需要改变复合条件的判断为一系列只有单个条件的嵌套的判断。例如，对应如图 6-13 所示的复合逻辑下的控制流图（a）的复合条件的判定，应该画成如图 6-13（b）所示的控制流图。条件语句 if a and b 中条件 a 和条件 b 各有一个只有单个条件的判断结点。

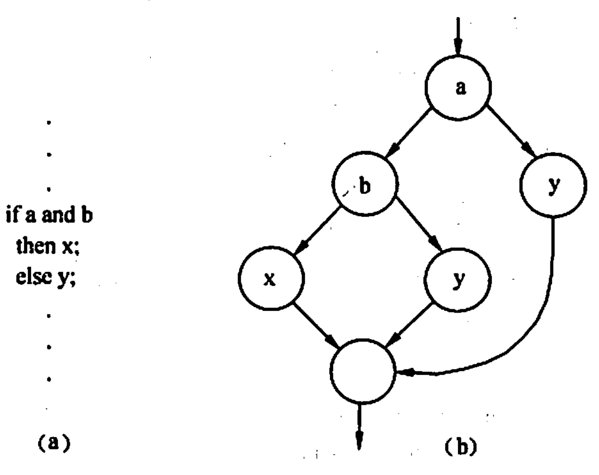


图 6-13　复合逻辑下的控制流图

2．程序环路复杂性

程序的环路复杂性即 McCabe 复杂性度量，在进行程序的基本路径测试时，从程序的环路复杂性可导出程序基本路径集合中的独立路径条数，这是确保程序中每个可执行语句至少执行一次所必须的测试用例数目的上界。

独立路径是指包括一组以前没有处理的语句或条件的一条路径。从控制流图来看，一条独立路径是至少包含有一条在其他独立路径中从未有过的边的路径。例如，在如图 6-12（b）中所示的控制流图中，一组独立的路径如下：

path1：1—11；

path2：1—2—3—4—5—10—1—11；

path3：1—2—3—6—8—9—10—1—11；

path4：1—2—3—6—7—9—10—1—11。



从此例中可知，一条新的路径必须包含有一条新边。路径 1—2—3—4—5—10—1—2—3—6—8—9—10—1—11 不能作为一条独立路径，因为它只是前面已经说明了的路径的组合，没有通过新的边。

路径 path1、path2、path3 和 path4 组成了如图 6-12（b）所示控制流图的一个基本路径集。只要设计出的测试用例能够确保这些基本路径的执行，就可以使得程序中的每个可执行语句至少执行一次，每个条件的取真和取假分支也能得到测试。基本路径集不是惟一的，对于给定的控制流图，可以得到不同的基本路径集。

通常环路复杂性还可以简单地定义为控制流图的区域数。这样对于如图 6-12（b）所示的控制流图，它有 4 个区域，环路复杂性 V（G）=4，它是构成基本路径集的独立路径数的上界，可以据此得到应该设计的测试用例的数目。

3．基本路径测试法步骤

基本路径测试法适用于模块的详细设计及源程序，其主要步骤如下：

- 以详细设计或源代码作为基础，导出程序的控制流图；
- 计算得到的控制流图 G 的环路复杂性 V（G）；
- 确定线性无关的路径的基本集；
- 生成测试用例，确保基本路径集中每条路径的执行。

下面以一个求平均值的过程 averagy 为例，说明测试用例的设计过程。用 PDL 语言描述的 averagy 过程如下所示。

```
PROCEDURE averagy;
* This procedure computes the average of 100 or fewer numbers that lie bounding values; it also
  computes the total input and the total valid.
  INTERFACE RETURNS averagy, total.input, total.valid;
  INTERFACE ACCEPTS value, minimum, maximum;
  TYPE value[1:100] IS SCALAR ARRAY;
  TYPE averagy, total.input, total.valid, minimum, maximum, sum IS SCALAR;
  TYPE i IS INTEGER;
  i = 1;
  total.input = total.valid = 0;
  sum = 0;
  DO WHILE value[i] <> -999 AND total.input < 100
      increment total.input by 1;
          IF value[i] >= minimum AND value[i] <= maximum
            THEN increment total.valid by 1;
                  sum = sum + value[i];
            ELSE skip;
```

```
      ENDIF;
     increment i by 1;
  ENDDO
  IF total.valid > 0
    THEN averagy = sum/total.valid;
    ELSE averagy = –999;
  ENDIF
END avergy
```

（1）以详细设计或源代码作为基础，导出程序的控制流图

利用在如图 6-11 所示的控制流图的图形符号、如图 6-12 所示的程序流图和对应的控制流图和如图 6-13 所示的复合逻辑下的控制流图给出的符号和构造规则生成控制流图。对于上述过程，对将要映射为对应控制流图中一个结点的 PDL 语句或语句组，加上用数字表示的标号。加了标号的 PDL 程序及对应的控制流图如图 6-14 和图 6-15 所示。

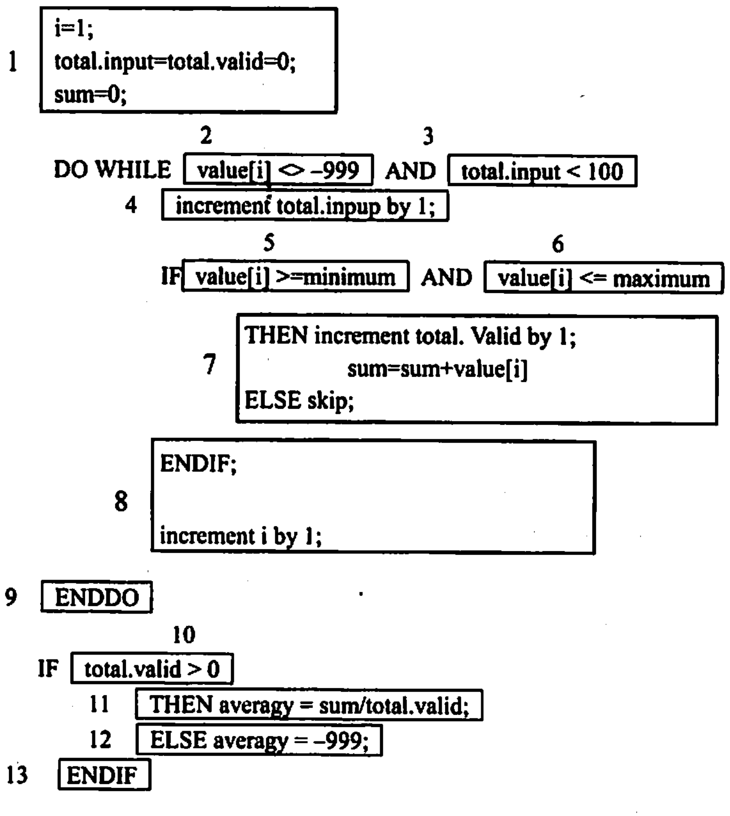


图 6-14　对 averagy 过程定义结点

（2）计算得到的控制流图 G 的环路复杂性 V（G）

利用在前面给出的计算控制流图环路复杂性的方法，算出控制流图 G 的环路复杂性。如果一开始就知道判断结点的个数，甚至不必画出整个控制流图，就可以计算出该图的环路复杂性的值。对于如图 6-15 所示的控制流图，可以算出：

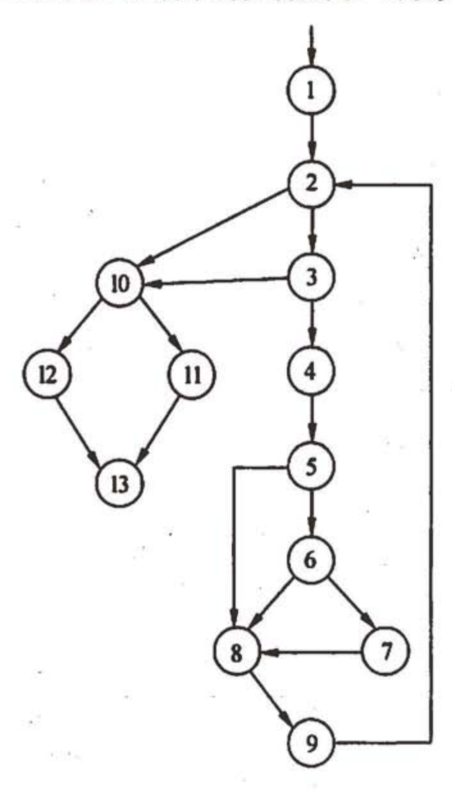


图 6-15 averagy 过程的控制流图

V(G)=6（区域数）=5（判断结点数）+1=6。

（3）确定线性无关路径的基本集

针对如图 6-15 所示的 averagy 过程的控制流图计算出的环路复杂性的值，就是该图已有的线性无关基本路径集中路径数目。该图所有的 6 条路径如下所示。

[path1]1－2－10－11－13

[path2]1－2－10－12－13

[path3]1－2－3－10－11－13

[path4]1－2－3－4－5－8－9－2…

[path5]1－2－3－4－5－6－8－9－2…

[path6]1－2－3－4－5－6－7－8－9－2…

路径 4、5、6 后面的省略号（…）表示在控制结构中以后剩下的路径是可选择的。在很多情况下，标识判断结点，常常能够有效地帮助导出测试用例。在上例中，结点 2、3、5、6 和 10 都是判断结点。

（4）生成测试用例，确保基本路径集中每条路径的执行

根据判断结点给出的条件，选择适当的数据以保证某一条路径可以被测试到。满足上述基本路径集的测试用例如下所示。

[path1]　输入数据：value[k]＝有效输入，限于 k<i（i 定义如下）
　　　　value[i]＝–999，当 2≤i≤100。
　　预期结果：n 个值的正确的平均值、正确的总计数。
　　注意：不能孤立地进行测试，应当作为路径 4、5、6 测试的一部分来测试。

[path2]　输入数据：value[1]＝–999；
　　预期结果：平均值=–999，总计数取初始值。

[path3]　输入数据：试图处理 101 个或更多的值，而前 100 个应当是有效的值；
　　预期结果：与测试用例 1 相同。

[path4]　输入数据：value[i]＝有效输入，且 i<100；
　　　　value[k]<最小值，当 k<i 时；
　　预期结果：n 个值的正确的平均值、正确的总计数。

[path5]　输入数据：value[i]＝有效输入，且 i<100；
　　　　value[k]>最大值，当 k≤i 时；
　　预期结果：n 个值的正确的平均值、正确的总计数。

[path6]　输入数据：value[i]＝有效输入，且 i<100
　　预期结果：n 个值的正确的平均值、正确的总计数。

每个测试用例执行之后，与预期结果进行比较。如果所有测试用例都执行完毕，则可以确信程序中所有的可执行语句至少被执行了一次。但是必须注意的是，一些独立的路径（如此例中的路径 1），往往不是完全孤立的，有时它是程序正常的控制流的一部分，这时，这些路径的测试可以是另一条路径测试的一部分。

6.2.6　其他白盒测试方法

1．域测试

域测试（Domain Testing）是一种基于程序结构的测试方法。Howden 曾对程序中出现的错误进行分类，他将程序错误分为域错误、计算型错误和丢失路径错误三种，这是相对于执行程序的路径来说的。我们知道，每条执行路径对应于输入域的一类情况，是



程序的一个子计算。如果程序的控制流有错误，对于某一特定的输入可能执行的是一条错误路径，这种错误称为路径错误，也叫做域错误。如果对于特定输入执行的是正确路径，但由于赋值语句的错误致使输出结果不正确，则称此为计算型错误。

另外一类错误是丢失路径错误。它是由于程序中的某处少了一个判定谓词而引起的。域测试主要是针对域错误进行的程序测试。

域测试的“域”是指程序的输入空间。域测试方法基于对输入空间的分析。自然，任何一个被测程序都有一个输入空间。测试的理想结果就是检验输入空间中的每一个输入元素是否都产生正确的结果。而输入空间又可分为不同的子空间，每一子空间对应一种不同的计算。在考察被测试程序的结构以后，我们就会发现，子空间的划分是由程序中分支语句中的谓词决定的。输入空间的一个元素，经过程序中某些特定语句的执行而结束（当然也可能出现无限循环而无出口），那都是满足了这些特定语句被执行所要求的条件的。

域测试正是在分析输入域的基础上，选择适当的测试点以后进行测试的。

域测试有两个致命的弱点，一是为进行域测试对程序提出的限制过多，二是当程序存在很多路径时，所需的测试点也就很多。

2．符号测试

符号测试的基本思想是允许程序的输入不仅仅是具体的数值数据，而且包括符号值，这一方法也是因此而得名的。这里所说的符号值可以是基本符号变量值，也可以是这些符号变量值的一个表达式。这样，在执行程序过程中以符号的计算代替了普通测试执行中对测试用例的数值计算。所得到的结果自然是符号公式或是符号谓词。更明确地说，普通测试执行的是算术运算，符号测试执行的则是代数运算。因此符号测试可以认为是普通测试的一个自然的扩充。

符号测试可以看作是程序测试和程序验证的一个折衷方法。一方面，它沿用了传统的程序测试方法，通过运行被测程序来验证它的可靠性。另一方面，由于一次符号测试的结果代表了一大类普通测试的运行结果，实际上是证明了程序接受此类输入后，所得输出是正确的还是错误的。最为理想的情况是，程序中仅有有限的几条执行路径。如果对这有限的几条路径都完成了符号测试，我们就能较有把握地确认程序的正确性了。

从符号测试方法的使用来看，问题的关键在于开发出比传统的编译器功能更强，能够处理符号运算的编译器和解释器。

目前符号测试存在一些未得到圆满解决的问题，如下所示。

- 分支问题。

当采用符号执行方法进行到某一分支点处，分支谓词是符号表达式，这种情况下通常无法决定谓词的取值，也就不能决定分支的走向，需要测试人员做人工干预，或是执行树的方法进行下去。如果程序中有循环，而循环次数又决定于输入变量，那就无法确

定循环的次数。

- 二义性问题。

数据项的符号值可能是有二义性的。这种情况通常出现在带有数组的程序中。

我们来看以下的程序段：

```
    …
X(I)=2+A
  X(J)=3
  C=X(I)
    …
```

如果 I=J，则 C=3；否则，C=2+A。但由于使用符号值运算，这时无法知道 I 是否等于 J。

- 大程序问题。

符号测试中经常要处理符号表达式。随着符号执行的继续，一些变量的符号表达式会越来越庞大。特别是当符号执行树很大，分支点很多时，路径条件本身变成一个非常长的表达式。如果能够有办法将其化简，自然会带来很大好处。但如果找不到化简的办法，那将给符号测试的时间和运行空间带来大幅度的增长，甚至使整个问题的解决遇到难以克服的困难。

3. Z 路径覆盖

分析程序中的路径是指：检验程序从入口开始，执行过程中经历的各个语句，直到出口。这是白盒测试最为典型的问题。着眼于路径分析的测试可称为路径测试。完成路径测试的理想情况是做到路径覆盖。对于比较简单的小程序实现路径覆盖是可能做到的。但是如果程序中出现多个判断和多个循环，可能的路径数目将会急剧增长，达到天文数字，以至于实现完全路径覆盖是不可能做到的。

为了解决这一问题，我们必须舍掉一些次要因素，对循环机制进行简化，从而极大地减少路径的数量，使得覆盖这些有限的路径成为可能。我们称简化循环意义下的路径覆盖为 Z 路径覆盖。

这里所说的对循环化简，是指限制循环的次数。无论循环的形式和实际执行循环体的次数多少，我们只考虑循环一次和零次两种情况。即只考虑执行时进入循环体一次和跳过循环体这两种情况。如图 6-16（a）和（b）所示表示了两种最典型的循环控制结构。前者先作判断，循环体 B 可能执行（假定只执行一次），也可能不执行。这就如同如图 6-16（c）所示的条件选择结构一样。后者先执行循环体 B（假定也执行一次），再经判断转出，其效果也与（c）中给出的条件选择结构只执行右支的效果一样。



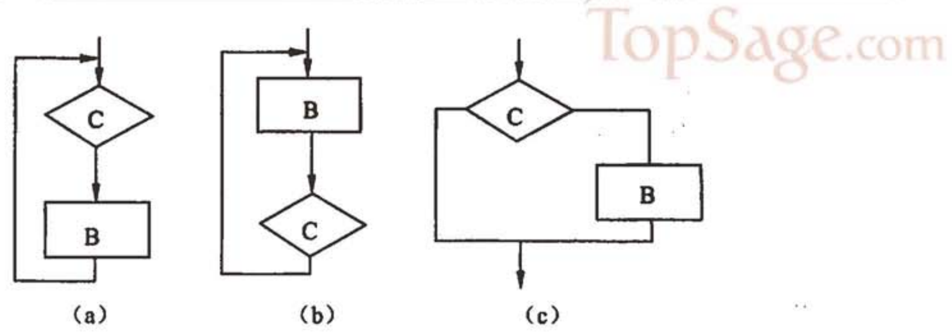


图 6-16 循环结构简化成选择结构

对于程序中的所有路径可以用路径树来表示，具体表示方法这里忽略。当得到某一程序的路径树后，从其根结点开始，一次遍历，再回到根结点时，把所经历的叶结点名排列起来，就得到一个路径。如果我们设法遍历了所有的叶结点，就得到了所有的路径。

当得到所有的路径后，生成每个路径的测试用例，就可以做到 Z 路径覆盖测试。

4. 程序变异

程序变异方法与前面提到的结构测试和功能测试都不一样，它是一种错误驱动测试。所谓错误驱动测试，是指该方法是针对某类特定程序错误的。经过多年的测试理论研究和软件测试的实践，人们逐渐发现要想找出程序中所有的错误几乎是不可能的。比较现实的解决办法是将错误的搜索范围尽可能地缩小，以利于专门测试某类错误是否存在。这样做的好处在于，便于集中目标于对软件危害最大的可能错误，而暂时忽略对软件危害较小的可能错误。这样可以取得较高的测试效率，并降低测试的成本。

错误驱动测试主要有两种，即程序强变异和程序弱变异。为便于测试人员使用变异方法，一些变异测试工具被开发出来。关于程序变异测试方法，请参见清华大学出版社出版的《软件测试技术》一书。

6.3 白盒测试综合策略

在白盒测试中，可以使用各种测试方法的综合策略如下所示。

- 在测试中，应尽量先用工具进行静态结构分析。
- 测试中可采取先静态后动态的组合方式：先进行静态结构分析、代码检查和静态质量度量，再进行覆盖率测试。
- 利用静态分析的结果作为引导，通过代码检查和动态测试的方式对静态分析结果进行进一步的确认，使测试工作更为有效。
- 覆盖率测试是白盒测试的重点，一般可使用基本路径测试法达到语句覆盖标

准；对于软件的重点模块，应使用多种覆盖率标准衡量代码的覆盖率；具体的测试用例数计算与覆盖准则参见本书后面的内容。

- 在不同的测试阶段，测试的侧重点不同：在单元测试阶段，以代码检查、逻辑覆盖为主；在集成测试阶段，需要增加静态结构分析、静态质量度量；在系统测试阶段，应根据黑盒测试的结果，采取相应的白盒测试。

6.3.1　最少测试用例数计算

为实现测试的逻辑覆盖，必须设计足够多的测试用例，并使用这些测试用例执行被测程序，实施测试。我们关心的是，对某个具体程序来说，至少要设计多少测试用例。这里提供一种估算最少测试用例数的方法。

我们知道，结构化程序是由 3 种基本控制结构组成的。这 3 种基本控制结构就是：

- 顺序型——构成串行操作；
- 选择型——构成分支操作；
- 重复型——构成循环操作。

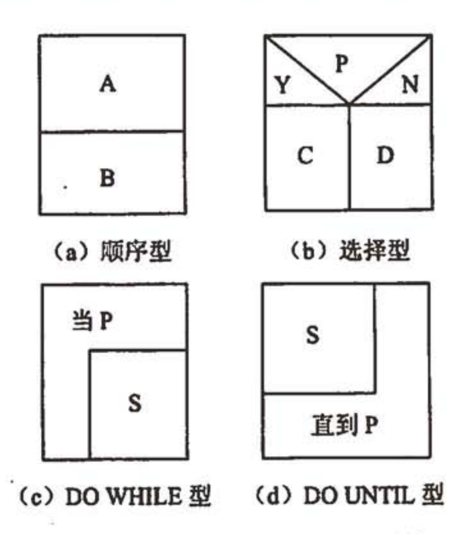
为了把问题化简，避免出现测试用例极多的组合爆炸，把构成循环操作的重复型结构用选择结构代替。也就是说，并不指望测试循环体所有的重复执行，而是只对循环体检验一次。这样，任一循环便改造成进入循环体或不进入循环体的分支操作了。

如图 6-17 所示给出了类似于流程图的 N-S 图表示的基本控制结构（图中 A、B、C、D、S 均表示要执行的操作，P 是可取真假值的谓词，Y 表真值，N 表假值）。其中图 6-17（c）和图 6-17（d）两种重复型结构代表了两种循环。在作了如上简化循环的假设以后，对于一般的程序控制流，我们只考虑选择型结构。事实上它已经能体现顺序型和重复型结构了。

例如，如图 6-18 所示表达了两个顺序执行的分支结构。两个分支谓词 P1 和 P2 取不同值时，将分别执行 a 或 b 及 c 或 d 操作。显然，要测试这个小程序，需要至少提供 4 个测试用例才能做到逻辑覆盖。使得 ac、ad、bc 及 bd 操作均得到检验。其实，这里的 4 是图中第 1 个分支谓词引出的两个操作，及第 2 个分支谓词引出的两个操作组合起来而得到的，即 $2\times2=4$。并且，这里的 2 是由于两个并列的操作，$1+1=2$ 而得到的。

对于一般的、更为复杂的问题，估算最少测试用例数的原则也是同样的。现以图 6-19 所示的程序为例。该程序中共有 9 个分支谓词，尽管这些分支结构交错起来似乎十分复杂，很难一眼看出应至少需要多少个测试用例，但如果仍用上面的方法，也是很容易解决的。我们注意到该图可分上下两层：分支谓词 1 的操作域是上层，分支谓词 8 的操作域是下层。这两层正像前面简单例中的 P1 和 P2 的关系一样。只要分别得到两层的测试用例个数，再将其相乘，即得总的测试用例数。这里需要首先考虑较为复杂的上层结构。

谓词 1 不满足时要做的操作又可进一步分解为两层，这就是如图 6-20 所示中的图（a）和（b）。它们所需测试用例个数分别为 1+1+1+1+1 = 5 和 1+1+1 = 3。因而两层组合，得到 5×3 = 15。于是整个程序结构上层所需测试用例数为 1+15 = 16，而下层十分显然为 3。故最后得到整个程序所需测试用例数至少为 16×3 = 48。



（a）顺序型　（b）选择型

（c）DO WHILE 型　（d）DO UNTIL 型

图 6-17　N-S 图表示的基本控制结构

图 6-18　两个串行的分支结构的 N-S 图

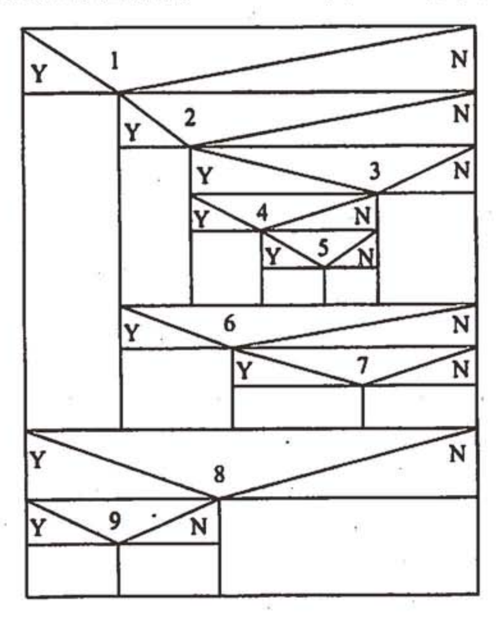


图 6-19　计算最少测试用例数实例

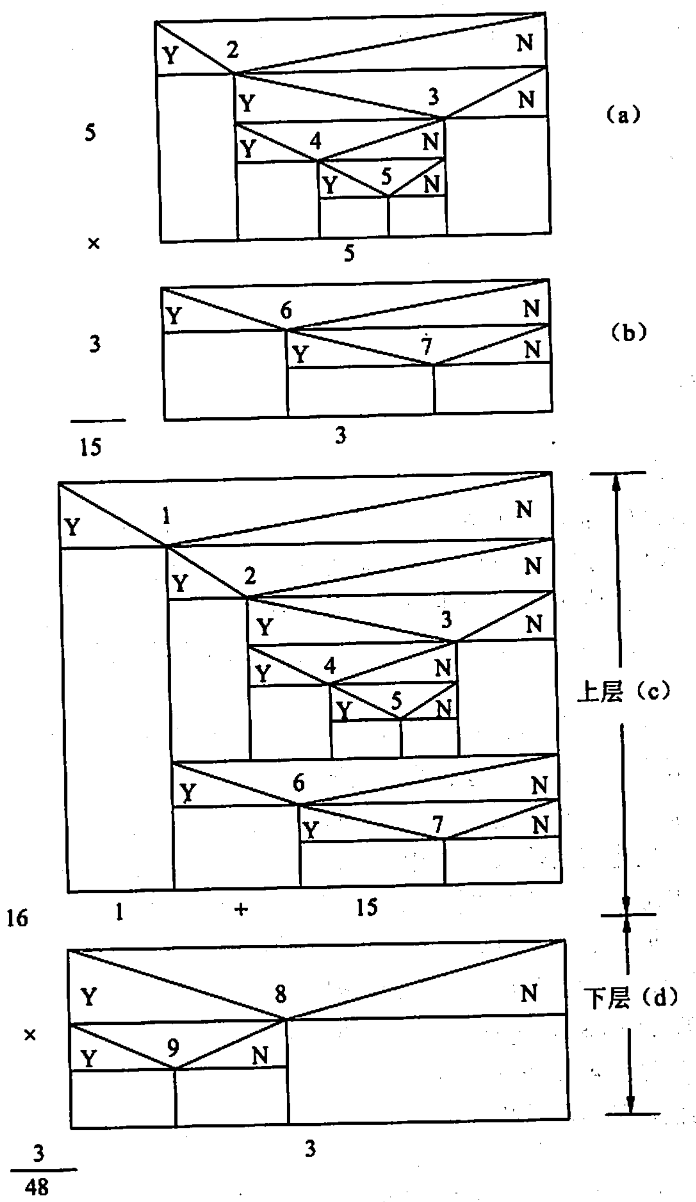


图 6-20　最少测试用例数计算

6.3.2　测试覆盖准则

1. Foster 的 ESTCA 覆盖准则

前面介绍的逻辑覆盖，其出发点似乎是合理的。所谓"覆盖"，就是想要做到全面



而无遗漏。但事实表明，它并不能真的做到无遗漏。甚至对于像前面提到的，将程序段：

```
…
IF(I ≥ 0)
THENI = J
…
```

错写成：

```
…
IF(I > 0)
THENI = J
…
```

这样的小问题都无能为力。

我们分析出现这一情况的原因在于，错误区域仅仅在 I = 0 这个点上，即仅当 I 取 0 时，测试才能发现错误。它的确是在我们力图通过全面覆盖来查找错误的测试的“网上”钻了空子，并且恰恰在容易发生问题的条件判断那里未被发现。面对这类情况，我们应该从中吸取的教训是测试工作要有重点，要多针对容易发生问题的地方设计测试用例。

K.A.Foster 从测试工作实践的教训出发，吸收了计算机硬件的测试原理，提出了一种经验型的测试覆盖准则，较好地解决了上述问题。

Foster 的经验型覆盖准则是从硬件的早期测试方法中得到启发的。我们知道，硬件测试中，对每一个门电路的输入、输出测试都是有额定标准的。通常，电路中一个门的错误常常是“输出总是 0”，或是“输出总是 1”。与硬件测试中的这一情况类似，我们常常要重视程序中谓词的取值，但实际上它可能比硬件测试更加复杂。Foster 通过大量的实验确定了程序中谓词最容易出错的部分，得出了一套错误敏感测试用例分析 ESTCA（Error Sensitive Test Cases Analysis）规则。事实上，规则十分简单。

- [规则 1] 对于 A rel B（rel 可以是<, = 和>）型的分支谓词，应适当地选择 A 与 B 的值，使得测试执行到该分支语句时，A<B, A = B 和 A>B 的情况分别出现一次。
- [规则 2] 对于 A rel1 C（rel1 可以是“>”或是“<”，A 是变量，C 是常量）型的分支谓词，当 rel1 为“<”时，应适当地选择 A 的值，使：

$$A = C - M。$$

（M 是距 C 最小的容器容许的正数，若 A 和 C 均为整型时，M = 1）。同样，当 rel1 为“>”时，应适当地选择 A，使：

A = C + M。

- [规则 3] 对外部输入变量赋值，使其在每一测试用例中均有不同的值和符号，并与同一组测试用例中其他变量的值和符号不一致。

显然，上述规则 1 是为了检测 rel 的错误，规则 2 是为了检测“差 1”之类的错误（如本应是“IF A>1”而错成“IF A>0”），而规则 3 则是为了检测程序语句中的错误（如应引用一变量而错成引用一常量）。

上述三个规则并不完备，但在普通程序的测试中确是有效的。原因在于规则本身针对着程序编写人员容易发生的错误，或是围绕着发生错误的频繁区域，从而提高了发现错误的命中率。

根据这里提供的规则来检验上述小程序段错误。应用规则 1，对它测试时，应选择 I 的值为 0，使 I = 0 的情况出现一次。这样一来就立即找出了隐藏的错误。

当然，ESTCA 规则也有很多缺陷。一方面是，有时不容易找到输入数据使得规则所指的变量值满足要求。另一方面是，仍有很多缺陷发现不了。对于查找错误的广度问题在变异测试中得到了较好的解决。

2．Woodward 等人的层次 LCSAJ 覆盖准则

Woodward 等人曾经指出结构覆盖的一些准则，如分支覆盖或路径覆盖，都不足以保证测试数据的有效性。为此，他们提出了一种层次 LCSAJ 覆盖准则。

LCSAJ(Linear Code Sequence and Jump)的意思是线性代码序列与跳转。一个 LCSAJ 是一组顺序执行的代码，以控制流跳转为其结束点。它不同于判断—判断路径。判断—判断路径是根据程序有向图决定的。一个判断—判断路径是指两个判断之间的路径，但其中不再有判断。程序的入口、出口和分支结点都可以是判断点。而 LCSAJ 的起点是根据程序本身决定的。它的起点是程序第一行或转移语句的入口点，或是控制流可以跳达的点。几个首尾相接，且第一个 LCSAJ 起点为程序起点，最后一个 LCSAJ 终点为程序终点的 LCSAJ 串就组成了程序的一条路径。一条程序路径可能是由两个、三个或多个 LCSAJ 组成的。基于 LCSAJ 与路径的这一关系，Woodward 提出了 LCSAJ 覆盖准则。这是一个分层的覆盖准则，具体如下。

- [第一层]：语句覆盖。
- [第二层]：分支覆盖。
- [第三层]：LCSAJ 覆盖。即程序中的每一个 LCSAJ 都至少在测试中经历过一次。
- [第四层]：两两 LCSAJ 覆盖。即程序中每两个首尾相连的 LCSAJ 组合起来在测试中都要经历一次。
- …
- [第 n+2 层]：每 n 个首尾相连的 LCSAJ 组合在测试中都要经历一次。

它们说明了，越是高层的覆盖准则越难满足。

在实施测试时，若要实现上述的 Woodward 层次 LCSAJ 覆盖，需要产生被测程序的所有 LCSAJ。

6.4 结论

从上面的概念和例子可以看出，要进行上面的白盒测试是需要投入巨大的测试资源，包括人力、物力和时间等。但是为什么还要进行白盒测试呢？原因如下。

- 逻辑错误和不正确假设与一条程序路径被运行的可能性成反比。当我们设计和实现主流之外的功能、条件或控制时，错误往往开始出现在我们的工作中。日常处理往往被很好地了解（和很好地细查），而“特殊情况”的处理则难以发现。
- 我们经常相信某逻辑路径不可能被执行，而事实上，它可能在正常的基础上被执行。程序的逻辑流有时是违反直觉的，这意味着我们关于控制流和数据流的一些无意识的假设，可能导致设计错误。只有路径测试才能发现这些错误。
- 印刷上的错误是随机的。当一个程序被翻译为程序设计语言源代码时，有可能产生某些打印错误，很多将被语法检查机制发现，但是，其他的错误只有在测试开始时才会被发现。打印错误出现在主流上和出现在不明显的逻辑路径上的可能性是一样的。

第 7 章　面向对象的软件测试技术

7.1　面向对象测试概述

对象概念对软件开发具有极大的好处，用户应用 OOP（面向对象编程）技术可以只编写一次代码而在今后反复重用，而在非 OOP 的情况下，则多半要在应用程序内部各个部分反复多次地编写同样的功能代码。所以，OOP 减少了编写代码的总量，加快了开发的进度。

但是面向对象编程也存在一些固有的缺点。例如，某个类被修改了，那么所有依赖该类的代码都必须重新测试，而且，还可能需要重新修改依赖类以支持类的变更。另外，如果相关开发文档没有得到仔细的维护，那么就很难确定哪些代码采用了父类（被继承的代码）。而且，如果在开发后期发现了软件中的错误，它很可能影响应用程序中的绝大部分的代码。

尽管面向对象技术的基本思想似乎表明了，这种技术开发出来的软件应该有更高的质量，但实际情况却并非如此，因为无论采用什么样的编程技术，编程人员的错误都是不可避免的，而且由于面向对象技术开发的软件代码重用率高，更需要严格的测试，避免错误的繁衍。因此，软件测试并没有因面向对象编程的兴起而丧失掉它的重要性，相反，更加迫切地需要一些新的测试理念和测试方法。本章将会对面向对象的测试做一个深入的介绍。

7.2　面向对象技术

在介绍面向对象软件测试之前，有必要先对面向对象技术的基本概念做一下简单介绍。

7.2.1　对象和类

面向对象其实是现实世界模型的自然延伸。现实世界中任何实体都可以看作是对象。对象之间通过消息相互作用。另外，现实世界中任何实体都可归属于某类事物，任何对象都是某一类事物的实例。如果说传统的过程式编程语言是以过程为中心，以算法



为驱动的话，那么面向对象的编程语言则是以对象为中心，以消息为驱动的。用公式表示，过程式编程语言为：程序=算法+数据；面向对象编程语言为：程序=对象+消息。

在面向对象的程序设计中，类是其设计的核心，它实际是一种新的数据类型，也是实现抽象类型的工具，类是通过抽象数据类型的方法来实现的一种数据类型。类是对某一类对象的抽象；而对象是某一种类的实例，因此，类和对象是密切相关的。没有脱离对象的类，也没有不依赖于类的对象。

7.2.2 封装、继承和多态性

封装，把数据和操作结合成一体，使程序结构更加紧凑，同时避免了数据紊乱带来的调试与维护的困难；继承，增加了软件的可扩充性，并为代码重用提供了强有力的手段；多态性，使程序员在设计程序时可以对问题进行更好的抽象，易设计出重用性和维护性俱佳的程序。

封装是将数据和操作数据的函数衔接在一起，构成的一个具有类类型的对象的描述。封装要求一个对象应具备明确的功能，并且有一个或几个接口以便和其他对象相互作用。同时，对象的内部实现（代码和数据）是受保护的，外界不能访问它们，只有对象中的代码才可以访问该对象的内部数据。对象的内部数据结构的不可访问性称为数据隐藏。封装简化了程序员对对象的使用，只需要知道输入什么和输出什么，而对类内部进行什么操作不必追究。

例如，类 Test（以 C++为例）：

```
#include<math.h>
#include<iostream.h>
class test
        {
Public:
int GetValue()
            {
SetValue();
Value+=8;
return Value;
            };
Protected:
void SetValue();
            {
Value=6;
```

```
            };
Private:
Int Value;    //数据是私有的
        };

void main()
        {
Test test;    //定义 test 对象
int value1;
double value2;
value1=test.GetValue();
value2=value1+2;
cout<<value2<<"\n";
        }
```

输出结果是 16，对于 main()里的操作，无法更改类 Test 中的 Value 值，这样保护了数据 Value，实现了数据的隐藏。函数 SetValue() 和数据 Value 结合在一起操作，这就实现了封装。

继承即可以从一个类派生另一个类。派生类（也称为子类）继承了其父类和祖先类的数据成员和成员函数。派生类可增加新的属性和新的操作，另外，派生类在继承的成员函数不合适时也可以放弃不用。

例如，水果是基类，它具有好吃、营养等水果的基本特征；香蕉和苹果是水果的派生类，它们在继承水果类的一些特征的基础上，增加了各自的特征，如特殊的味道、形状等；红富士苹果和青苹果是苹果类的派生类，它们又增加了各自的新特征。如此继承设计，可以大大减少工作量。

例如（C++程序），先设计水果类 Friut。

```
Class Friut
        {
public:
void Eat(){cout<<" Taste is good.\n";}
void Taste(){count<<" Can be eaten.\n";}
        }
```

苹果类 Apple 需要与水果类 Friut 同样的成员函数 void Eat 和 void Taste，另外增加了新的函数 void Color 和 void Shape。在设计时，我们不需要重写 void Eat 和 void Taste，只需要继承水果类即可，苹果类的实际代码如下。



```
class Apple:public Friut
            {
public:
void Color(){cout<<" Diffent kind of color.\n";}
void Shape(){count<<" The shape is circle.\n";}
            }
```

多态性就是多种表现形式，具体来说，可以用“一个对外接口，多个内在实现方法”表示。多态性的实现，一般通过在派生类中重定义基类的虚函数来实现。

如下例（C++程序），如果 A 是基类，B 和 C 是 A 的派生类，基类中多态函数 Test 的参数是 A 的指针。那么 Test 函数可以引用 A、B、C 的对象如下。

```
class A
            {
Public:
virtual void Func1(void){ cout<< "This is A::Func1 \n"}
            };

void Test(A *a)
            {
a->Func1();
            }

class B : public A
            {
Public:
virtual void Func1(void){ cout<< "This is B::Func1 \n"}
            };
class C : public A
            {
Public:
virtual void Func1(void){ cout<< "This is C::Func1 \n"}
            };

// Example
main()
            {
```

```
A a;
B b;
C c;
        Test(&a);   // 输出 This is A::Func1
        Test(&b);   // 输出 This is B::Func1
        Test(&c);   // 输出 This is C::Func1
        };
```

正是面向对象软件所独有的这些多态、继承、封装等新特点，使 OO 程序设计比传统语言程序设计产生错误的可能性更大，使得传统软件测试中的重点不再显得那么突出，也使原来测试经验和实践证明的次要方面成为了主要问题。例如，在传统的面向过程程序中，对于函数 y=Func (x)；你只需要考虑一个函数“Func()”的行为特点，而在面向对象程序中，你不得不同时考虑基类函数“Base::Func()”的行为和继承类函数“Derived::Func()”的行为。

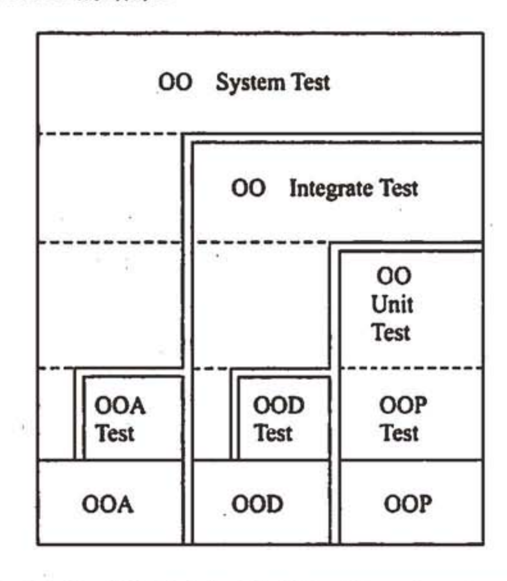
面向对象程序的结构不再是传统的功能模块结构，作为一个整体，原有集成测试所要求的，逐步将开发的模块搭建在一起进行测试的方法已不可能。而且，面向对象软件抛弃了传统的开发模式，对每个开发阶段都有不同于以往的要求和结果，已经不可能用功能细化的观点来检测面向对象分析和设计的结果。因此，传统的测试模型对面向对象软件已经不再适用。针对面向对象软件的开发特点，一种新的面向对象测试模型应需而生。

7.3　面向对象测试模型

面向对象的开发模型突破了传统的瀑布模型，将开发分为面向对象分析（OOA），面向对象设计（OOD）和面向对象编程（OOP）三个阶段。针对这种开发模型，结合传统的测试步骤的划分，我们把面向对象的软件测试分为：面向对象分析的测试、面向对象设计的测试、面向对象编程的测试、面向对象单元测试、面向对象集成测试、面向对象确认和系统测试。建议是一种整个软件开发过程中不断测试的测试模型，使开发阶段的测试与编码完成后的单元测试、集成测试、系统测试成为一个整体。测试模型如图 7-1 所示。

OOA Test 和 OOD Test 是对分析结果和设计结果的测试，主要是对分析设计产生的文本进行的，是软件开发前期的关键性测试。OOP Test 主要针对编程风格和程序代码实现进行测试，其主要的测试内容在面向对象单元测试和面向对象集成测试中体现。面向对象单元测试是对程序内部具体单一的功能模块的测试，主要就是对类成员函数的测

试。面向对象单元测试是进行面向对象集成测试的基础。面向对象集成测试主要对系统内部的相互服务进行测试，如成员函数间的相互作用，类间的消息传递等。面向对象集成测试不但要基于面向对象单元测试，更要参考 OOD 或 OOD Test 结果。面向对象确认和系统测试是基于面向对象集成测试的最后阶段的测试，主要以用户需求为测试标准，需要借鉴 OOA 或 OOA Test 的结果。



OOA Test：面向对象分析的测试	OOD Test：面向对象设计的测试
OOP Test：面向对象编程的测试	OO Unit Test：面向对象单元测试
OO Integrate Test：面向对象集成测试	OO System Test：面向对象确认和系统测试

图 7-1 面向对象测试模型

尽管上述各阶段的测试构成一个相互作用的整体，但其测试的主体、方向和方法各有不同。

7.4 面向对象软件的测试策略

7.4.1 面向对象分析（OOA）的测试

传统的面向过程分析是一个功能分解的过程，是把一个系统看成可以分解的功能的集合。这种传统的功能分解分析法的着眼点在于，一个系统需要什么样的信息处理方法

和过程，以过程的抽象来对待系统的需要。而面向对象分析（OOA）是“把 E-R 图和语义网络模型，即信息造型中的概念，与面向对象程序设计语言中的重要概念结合在一起而形成的分析方法”，最后通常是以问题空间的图表的形式进行描述。

OOA 直接映射问题空间，全面地将问题空间中实现功能的现实抽象化。将问题空间中的实例抽象为对象（不同于 C++中的对象概念），用对象的结构反映问题空间的复杂实例和复杂关系，用属性和服务表示实例的特性和行为。对一个系统而言，与传统分析方法产生的结果相反，行为是相对稳定的，结构是相对不稳定的，这更充分反映了现实的特性。OOA 的结果是为后面阶段类的选定和实现，类层次结构的组织和实现提供平台。因此，OOA 对问题空间分析抽象的不完整，最终会影响软件的功能实现，导致软件开发后期产生大量原本可避免的修补工作；另外，一些冗余的对象或结构会影响类的选定、程序的整体结构或增加程序员不必要的工作量。因此，对 OOA 测试的重点在其完整性和冗余性。

尽管 OOA 的测试是一个不可分割的系统过程，为叙述的方便，鉴于 Coad 方法所提出的 OOA 实现步骤，对 OOA 阶段的测试划分为以下五个方面：

- 对认定的对象的测试。
- 对认定的结构的测试。
- 对认定的主题的测试。
- 对定义的属性和实例关联的测试。
- 对定义的服务和消息关联的测试。

1．对认定的对象的测试

OOA 中认定的对象是对问题空间中的结构，其他系统、设备，被记忆的事件，系统涉及的人员等实际实例的抽象。对它的测试可以从如下方面考虑：

- 认定的对象是否全面，是否问题空间中所有涉及到的实例都反映在认定的抽象对象中了。
- 认定的对象是否具有多个属性。只有一个属性的对象通常应看成其他对象的属性，而不是抽象为独立的对象。
- 对认定为同一对象的实例是否有共同的、区别于其他实例的共同属性。
- 对认定为同一对象的实例是否提供或需要相同的服务，如果服务随着实例的不同而变化，认定的对象就需要分解或利用继承性来分类表示。
- 如果系统没有必要始终保持对象代表的实例的信息，提供或者得到关于它的服务、认定的对象也无必要。
- 认定的对象的名称应该尽量准确、适用。



2．对认定的结构的测试

在 Coad 方法中，认定的结构指的是多种对象的组织方式，用来反映问题空间中的复杂实例和复杂关系。认定的结构分为两种：分类结构和组装结构。分类结构体现了问题空间中实例的一般与特殊的关系，组装结构体现了问题空间中实例整体与局部的关系。

- 对认定的分类结构的测试可从如下方面着手：

① 对于结构中的一种对象，尤其是处于高层的对象，是否在问题空间中含有不同于下一层对象的特殊的可能性，即是否能派生出下一层对象。

② 对于结构中的一种对象，尤其是处于同一低层的对象，是否能抽象出在现实中有意义的更一般的上层对象。

③ 对所有认定的对象，是否能在问题空间内向上层抽象出在现实中有意义的对象。

④ 高层的对象的特性是否完全体现下层的共性。

⑤ 低层的对象是否有高层特性基础上的特殊性。

- 对认定的组装结构的测试从如下方面入手：

① 整体（对象）和部件（对象）的组装关系是否符合现实的关系。

② 整体（对象）的部件（对象）是否在考虑的问题空间中有实际应用。

③ 整体（对象）中是否遗漏了反映在问题空间中有用的部件（对象）。

④ 部件（对象）是否能够在问题空间中组装新的有现实意义的整体（对象）。

3．对认定的主题的测试

主题是在对象和结构的基础上更高一层的抽象，是为了提供 OOA 分析结果的可见性，如同文章对各部分内容的概要。对主题层的测试应该考虑以下方面：

- 贯彻 George Miller 的“7+2”原则（George Miller 指出普通人的短期记忆的能量，是 7 加 2 或减 2，即 5 与 9 之间）。如果主题个数超过 7 个，就要求对有较密切属性和服务的主题进行归并。
- 主题所反映的一组对象和结构是否具有相同和相近的属性和服务。
- 认定的主题是否是对象和结构更高层的抽象，是否便于理解 OOA 结果的概貌（尤其是对非技术人员的 OOA 结果读者）。
- 主题间的消息联系（抽象）是否代表了主题所反映的对象和结构之间的所有关联。

4．对定义的属性和实例关联的测试

属性是用来描述对象或结构所反映的实例的特性。而实例关联是反映实例集合间的映射关系。对属性和实例关联的测试从如下方面考虑：

- 定义的属性是否对相应的对象和分类结构的每个现实实例都适用。
- 定义的属性在现实世界是否与这种实例关系密切。

- 定义的属性在问题空间是否与这种实例关系密切。
- 定义的属性是否能够不依赖于其他属性被独立理解。
- 定义的属性在分类结构中的位置是否恰当，低层对象的共有属性是否在上层对象属性体现。
- 在问题空间中每个对象的属性是否定义完整。
- 定义的实例关联是否符合现实。
- 在问题空间中实例关联是否定义完整，特别需要注意“一对多”和“多对多”的实例关联。

5．对定义的服务和消息关联的测试

定义的服务，就是定义的每一种对象和结构在问题空间所要求的行为。由于问题空间中实例间必要的通信，在 OOA 中需要相应地定义消息关联。对定义的服务和消息关联的测试从如下方面进行：

- 对象和结构在问题空间的不同状态是否定义了相应的服务。
- 对象或结构所需要的服务是否都定义了相应的消息关联。
- 定义的消息关联所指引的服务提供是否正确。
- 沿着消息关联执行的线程是否合理，是否符合现实过程。
- 定义的服务是否重复，是否定义了能够得到的服务。

7.4.2　面向对象设计（OOD）的测试

在以往结构化的设计方法，用的“是面向作业的设计方法，它把系统分解以后，提出一组作业，这些作业是以过程实现系统的基础构造，把问题域的分析转化为求解域的设计，分析的结果是设计阶段的输入”。

而面向对象设计（OOD）采用“造型的观点”，以 OOA 为基础归纳出类，并建立类结构或进一步构造成类库，以实现分析结果对问题空间的抽象。OOD 归纳的类既可以是对象简单的延续，也可以是不同对象的相同或相似的服务。因此，OOD 是 OOA 的进一步细化和更高层的抽象。所以，OOD 与 OOA 的界限一般是很难严格区分的。OOD 确定类和类结构不仅是满足当前需求分析的要求，更重要的是通过重新组合或加以适当的补充，能方便实现功能的重用和扩增，以不断适应用户的要求。因此，对 OOD 的测试，建议针对功能的实现和重用以及对 OOA 结果的拓展，从如下三方面考虑：

- 对认定的类的测试。
- 对构造的类层次结构的测试。
- 对类库的支持的测试。



1．对认定的类的测试

OOD 认定的类可以是 OOA 中认定的对象，也可以是对象所需要的服务的抽象，对象所具有的属性的抽象。认定的类原则上应该尽量具有基础性，这样才便于维护和重用。测试认定的类包括如下方面：

- 是否涵盖了 OOA 中所有认定的对象。
- 是否能体现 OOA 中定义的属性。
- 是否能实现 OOA 中定义的服务。
- 是否对应着一个含义明确的数据抽象。
- 是否尽可能少地依赖其他类。
- 类中的方法（在 C++中即类的成员函数）是否是单用途。

2．对构造的类层次结构的测试

为了能够充分发挥面向对象的继承共享特性，OOD 的类层次结构，通常基于 OOA 中产生的分类结构的原则来组织，着重体现父类和子类间的一般性和特殊性。在当前的问题空间，对类层次结构的主要要求是能在解空间构造实现全部功能的结构框架。为此，需要测试如下方面：

- 类层次结构是否涵盖了所有定义的类。
- 是否能体现 OOA 中所定义的实例关联。
- 是否能实现 OOA 中所定义的消息关联。
- 子类是否具有父类没有的新特性。
- 子类间的共同特性是否完全在父类中得以体现。

3．对类库支持的测试

对类库的支持虽然也属于类层次结构的组织问题，但其强调的重点是再次软件开发的重用。由于它并不直接影响当前软件的开发和功能实现，因此，将其单独提出来测试，也可作为对高质量类层次结构的评估。拟定测试点如下：

- 一组子类中关于某种含义相同或基本相同的操作，是否有相同的接口（包括名字和参数表）。
- 类中方法（在 C++中即类的成员函数）的功能是否较单纯，相应的代码行是否较少（建议为不超过 30 行）。
- 类的层次结构是否是深度大，宽度小的。

因为 OOA、OOD 阶段所建立的分析和设计模型不能进行传统意义上的测试，它们不能被执行，所以在每次迭代之后，一定要进行评审。评审主要针对以下两个方面。

- 正确性

OO 开发模式为演化（重复迭代）性质，即系统的初期为非形式化表示，以后发展

为类的细节模型、类的连接和关联，系统设计和配置，以及对类的设计（通过消息组成对象连接模型）。每一阶段都要进行评审。

正确性主要在于分析和设计模型表示所使用的符号语法是否正确，语义是否正确（即模型与真实世界领域是否一致），以及类的关联（实例间的联系）是否正确地反映了真实世界对象间的关联。

- 一致性

由于演化性质，OOA 和 OOD 模型（包括分析、设计和编码层次，即类、属性、操作、消息）不仅要正确，而且要一致。一致性可以用模型内各实体间的关联性来判断。

为评估一致性，应检查每个类及其与其他类的连接。可运用类-责任-协作者（CRC）模型和对象-关系图。CRC 模型由 CRC 索引卡片构成，每个 CRC 卡片列出类名、类的责任（操作）、以及其协作者类，类向它们发送消息并依赖它们完成自己的责任。协作包含了在 OO 系统的类之间的一系列关系（即连接），对象关系模型提供了类之间连接的图形表示。所有这些信息可以从 OOA 模型得到。

为评估类模型，推荐采用以下步骤：

① 再次考察 CRC 模型和对象-关系模型，进行交叉检查，以保证由 OOA 模型所蕴含的协作适当地反应在二者中。

② 检查每个 CRC 索引卡片的描述，以确定是否某被授权的责任是协作者定义的一部分，例如，某个 POS 结账系统定义的类，称为 credit sale，该类的 CRC 卡片如图 7-2 所示。

对于这组类和协作，我们问如果某责任（如，read credit card）被委托给指定的协作者（credit card），该责任是否将被完成，即，类 credit card 是否具有一个操作以使得它可以被读。在本例的情形中，我们的回答是：“是”。遍历对象关系，以保证所有这样的连接是有效的。

③ 反转该连接，以保证每个被请求服务的协作者正在接收来自合理源的请求，例如，如果 credit card 类接收来自 credit sale 类的 purchase amount 请求，则将会有问题，因为 credit card 并不知道 purchase amount。

④ 使用在第③步检查的反转连接，确定是否可能需要其他的类，或责任是否被合适地在类间分组。

⑤ 确定是否被广泛请求的责任可被组合为单个的责任，例如，read credit card 和 get authorization 在每种情况均发生，它们可以被组合为 validate credit。request 责任，它结合了获取信用卡号及获得授权。

⑥ 步骤①～⑤被迭代地应用到每个类，并贯穿于 OOA 模型的每次演化过程中。

一旦已经创建了设计模型，也应该进行对系统设计和对象设计的复审。系统设计描



述了构成产品的子系统、子系统被分配到处理器的方式，以及类到子系统的分配。对象模型表示了每个类的细节和实现类间的协作所必须的消息序列活动。

类名：credit sale	
类的类型：交易事件	
类的特征：不确实的，原子的，顺序的，永久的，受保护的	
责任：	协作者：
read credit card	credit card
get authorization	credit authority
post purchase amount	product ticket
	sales ledger
	audit file
generate bill	bill

图 7-2　一个用于复审的 CRC 卡片例子

通过检查在 OOA 阶段开发的对象-行为模型，和映射需要的系统行为到被设计用于完成该行为的子系统来进行系统设计的复审。也在系统行为的语境内复审并发性和任务分配，评估系统的行为状态以确定哪些行为并发地存在。

对象模型应该针对对象-关系网络来测试，以保证所有设计对象包含为实现每张CRC索引卡片定义的协作所必须的属性和操作。

7.4.3　面向对象编程（OOP）的测试

面向对象程序所具有的继承、封装和多态的新特性，使得传统的测试策略不再完全适用。封装是对数据的隐藏，外界只能通过被提供的操作来访问或修改数据，这样降低了数据被任意修改和读写的可能性，降低了传统程序中对数据非法操作的测试。继承是面向对象程序的重要特点，继承使得代码的重用率提高，同时也使错误传播的概率提高。继承使得传统测试遇到了这样一个难题：对继承的代码究竟应该怎样测试？（参见面向对象单元测试）。多态使得面向对象程序对外呈现出强大的处理能力，但同时却使得程序内“同一”函数的行为复杂化，测试时不得不考虑不同类型具体执行的代码和产生的行为。

面向对象程序是把功能的实现分布在类中。能正确实现功能的类，通过消息传递来协同实现设计要求的功能。正是这种面向对象程序风格，将出现的错误能精确地确定在某一具体的类。因此，在面向对象编程（OOP）阶段，忽略类功能实现的细则，将测试的目光集中在类功能的实现和相应的面向对象程序风格上，主要体现为以下两个方面（假设编程使用 C++语言）。

- 数据成员是否满足数据封装的要求。
- 类是否实现了要求的功能。

1．数据成员是否满足数据封装的要求

数据封装是数据和数据有关的操作的集合。检查数据成员是否满足数据封装的要求，基本原则是数据成员是否被外界（数据成员所属的类或子类以外的调用）直接调用。更直观地说，当改变数据成员的结构时，是否影响了类的对外接口，是否会导致相应的外界必须改动。值得注意的是，有时强制的类型转换会破坏数据的封装特性。例如：

```
 class Hiden
 {private:
 int a=1;
 char *b= "hiden";}
class Visible
 {public:
 int c=2;
 char *d= "visible";}
 ⋮
 ⋮
 Hiden xx;
 Visible *yy=(Visible *)&xx;
```

在上面的程序段中，xx 的私有数据成员 a 和 b 可以通过 yy 被随意访问。

2．类是否实现了要求的功能

类所实现的功能都是通过类的成员函数执行的。在测试类的功能实现时，应该首先保证类成员函数的正确性。单独地看待类的成员函数，与面向过程程序中的函数或过程没有本质的区别，几乎所有传统的单元测试中所使用的方法，都可在面向对象的单元测试中使用。具体的测试方法在面向对象的单元测试中介绍。类函数成员的正确行为只是类能够实现要求的功能的基础，类成员函数间的作用和类之间的服务调用是单元测试无



法确定的。因此，需要进行面向对象的集成测试。具体的测试方法将在面向对象的集成测试中介绍。需要着重声明的是，测试类的功能，不能仅满足于代码能无错地运行或被测试类所提供的功能无错，应该以所做的 OOD 结果为依据，检测类提供的功能是否满足设计的要求，是否有缺陷。必要时还应该参照 OOA 的结果，以之为最终标准。

7.4.4 面向对象软件的单元测试

传统的单元测试是针对程序的函数、过程或完成某一特定功能的程序块，可沿用单元测试的概念，来实际测试类成员函数。一些传统的测试方法在面向对象的单元测试中都可以使用。如等价类划分法、因果图法、边值分析法、逻辑覆盖法、路径分析法、程序插装法等。单元测试一般建议由程序员完成。

用于单元级测试的测试分析（提出相应的测试要求）和测试用例（选择适当的输入，达到测试要求），规模和难度等均远小于后面将介绍的对整个系统的测试分析和测试用例，而且强调对语句应该有 100%的执行代码覆盖率。在设计测试用例选择输入数据时，可以基于以下两个假设（假设使用 C++编程语言）：

- 如果函数（程序）对某一类输入中的一个数据正确执行，对同类中的其他输入也能正确执行。
- 如果函数（程序）对某一复杂度的输入正确执行，对更高复杂度的输入也能正确执行。例如，需要选择字符串作为输入时，基于本假设，就无须计较字符串的长度。除非字符串的长度是要求固定的，如 IP 地址字符串。

在面向对象程序中，类成员函数通常都很小，功能单一，函数之间调用频繁，容易出现一些不宜发现的错误。例如：

```
if (-1= =write (fid, buffer, amount)) error_out();
```

该语句没有全面检查 write()的返回值，无意中只断然假设了数据被完全写入和没有写入两种情况。当测试也忽略了数据部分的写入情况时，就给程序遗留了隐患，建议加入 else 语句。

- 按程序的设计，使用函数 strrchr()查找最后的匹配字符，但错误程序中写成了函数 strchr()，使程序功能实现时查找的是第一个匹配字符。
- 程序中将 if (strncmp(str1,str2,strlen(str1)))误写成了 if (strncmp(str1,str2,strlen(str2)))。如果测试用例中使用的数据 str1 和 str2 长度一样，就无法检测出来。

因此，在做测试分析和设计测试用例时，应该注意面向对象程序的这个特点，仔细地进行测试分析和设计测试用例，尤其是针对以函数返回值作为条件判断选择，字符串操作等情况。

面向对象编程的特性使得对成员函数的测试，不完全等同于传统的函数或过程测试。尤其是继承特性和多态特性，使子类继承或过载的父类成员函数出现了传统测试中未遇见的问题。Brian Marick 给出了两方面的考虑如下：

- 继承的成员函数是否都不需要测试。

对父类中已经测试过的成员函数，有两种情况需要在子类中重新测试：

① 继承的成员函数在子类中做了改动；② 成员函数调用了改动过的成员函数的部分。例如：假设父类 Bass 有两个成员函数：Inherited()和 Redefined()，子类 Derived 只对 Redefined()做了改动。Derived::Redefined()显然需要重新测试。对于 Derived::Inherited()，如果它有调用 Redefined()的语句（如：x=x/Redefined()），就需要重新测试，反之，无此必要。

- 对父类的测试是否能照搬到子类。

延用上面的假设，Base::Redefined()和 Derived::Redefined()已经是不同的成员函数，它们有不同的服务说明和执行。对此，照理应该对 Derived::Redefined()重新进行测试分析，设计测试用例。但由于面向对象的继承使得两个函数有相似之处，故只需在 Base::Redefined()的测试要求和测试用例上添加对 Derived::Redfined()的新的测试要求和增补相应的测试用例。例如：Base::Redefined()函数中含有如下语句

```
If (value<0) message ("less");
else if (value==0) message ("equal");
else message ("more");
Derived::Redfined()中定义为
If (value<0) message ("less");
else if (value==0) message ("It is equal");
else
{message ("more");
if (value==99)message("luck");}
```

在原有的测试上，对 Derived::Redfined()的测试只需做如下改动：将 value==0 的测试结果期望改动；增加 value==99 的测试。

多态有几种不同的形式，如参数多态、包含多态、过载多态。包含多态和过载多态在面向对象语言中通常体现在子类与父类的继承关系上，对这两种多态的测试参见上述对父类成员函数继承和过载的论述。包含多态虽然使成员函数的参数可有多种类型，但通常只是增加了测试的复杂度。对具有包含多态的成员函数进行测试时，只需要在原有的测试分析和基础上增加对测试用例中输入数据的类型的考虑。



7.4.5 面向对象软件的集成测试

因为面向对象软件没有层次的控制结构，传统的自顶向下和自底向上集成策略就没有意义，而且，一次集成一个操作到类中（传统的增量集成方法）经常是不可能的，这是由于“构成类的成分的直接和间接的交互”。

此外，传统的自底向上通过集成完成的功能模块测试，一般可以在部分程序编译完成的情况下进行。而对于面向对象程序，相互调用的功能是散布在程序的不同类中的，类通过消息相互作用申请和提供服务。类的行为与它的状态密切相关，状态不仅仅是体现在类数据成员的值，也许还包括其他类中的状态信息。由此可见，类之间的相互依赖极其紧密，根本无法在编译不完全的程序上对类进行测试。所以，面向对象的集成测试通常需要在整个程序编译完成后进行。此外，面向对象程序具有动态特性，程序的控制流往往无法确定，因此，也只能对整个编译后的程序做基于黑盒子的集成测试。

对 OO 软件的集成测试有两种不同的策略，第一种称为基于线程的测试（thread based testing），集成对回应系统的一个输入或事件所需的一组类，每个线程被集成并分别测试，应用回归测试以保证没有产生副作用。第二种称为基于使用的测试（use based testing），通过测试那些几乎不使用服务器类的类（称为独立类）而开始构造系统，在独立类测试完成后，下一层中使用独立类的类（称为依赖类）被测试。这个依赖类层次的测试序列一直持续到构造完整个系统。序列和传统集成不同，使用驱动器和桩（stubs）作为替代操作是要尽可能避免的。

面向对象的集成测试，能够检测出相对独立的，单元测试无法检测出的，那些类相互作用时才会产生的错误。基于单元测试对成员函数行为正确性的保证，集成测试只关注于系统的结构和内部的相互作用。面向对象的集成测试可以分成两步进行：先进行静态测试，再进行动态测试。

静态测试主要针对程序的结构进行，检测程序结构是否符合设计要求。现在流行的一些测试软件都能提供一种称为“可逆性工程”的功能，即通过源程序得到类关系图和函数功能调用关系图，例如 International Software Automation 公司的 Panorama-2 for Windows 95、Rational 公司的 Rose C++ Analyzer 等，将“可逆性工程”得到的结果与 OOD 的结果相比较，检测程序结构和实现上是否有缺陷。换句话说，通过这种方法检测 OOP 是否达到了设计要求。

动态测试设计测试用例时，通常需要上述的功能调用结构图、类关系图或者实体关系图为参考，确定不需要被重复测试的部分，从而优化测试用例，减少测试工作量，使得进行的测试能够达到一定覆盖标准。测试所要达到的覆盖标准可以是：达到类所有的

服务要求或服务提供的一定覆盖率；依据类间传递的消息，达到对所有执行线程的一定覆盖率；达到类的所有状态的一定覆盖率等。同时也可以考虑使用现有的一些测试工具来得到程序代码执行的覆盖率。

具体设计测试用例，可参考下列步骤。

① 先选定检测的类，参考 OOD 分析结果，仔细确定出类的状态和相应的行为，类或成员函数间传递的消息，输入或输出的界定等。

② 确定覆盖标准。

③ 利用结构关系图确定待测类的所有关联。

④ 根据程序中类的对象构造测试用例，确认使用什么输入激发类的状态，使用类的服务和期望产生什么行为等。

值得注意的是，设计测试用例时，不但要设计确认类功能满足的输入，还应该有意识地设计一些被禁止的例子，确认类是否有不合法的行为产生，如发送与类状态不相适应的消息，要求不相适应的服务等。根据具体情况，动态的集成测试，有时也可以通过系统测试完成。

7.4.6 面向对象软件的确认和系统测试

通过单元测试和集成测试，仅能保证软件开发的功能得以实现，但不能确认在实际运行时，它是否能够满足用户的需要，是否大量地存在着实际使用条件下会被诱发产生错误的隐患。为此，对完成开发的软件必须进行规范的系统测试。即需要测试它与系统其他部分配套运行的表现，以确保在系统各部分协调工作的环境下软件也能正常运行。

系统测试应该尽量搭建与用户实际使用环境相同的测试平台，应该保证被测系统的完整性，对暂时没有的系统设备部件，应采取相应的模拟手段。系统测试时，应该参考 OOA 分析的结果，对应描述的对象、属性和各种服务，检测软件是否能够完全“再现”问题空间。系统测试不仅是检测软件的整体行为表现，从另一个侧面看，也是对软件开发设计的再确认。

在系统层次，类连接的细节消失了。和传统有效性测试一样，OO 软件的有效性集中在用户可见的动作和用户可识别的系统输出上。为了协助有效性测试的导出，测试人员应该利用作为分析模型一部分的使用实例，使用实例提供了在用户交互需求中很可能发现错误的一个场景。传统的黑盒测试方法可被用于驱动有效性测试，此外，测试用例可以从对象-行为模型和作为 OOA 的一部分的事件流图中导出。系统测试需要对被测的软件结合需求分析做仔细的测试分析，建立测试用例。

OO 软件确认和系统测试具体的测试内容与传统系统测试基本相同，包括：功能测试、强度测试、性能测试、安全测试、恢复测试、易用性测试、安装/卸载测试（install/uninstall

test）等，此处不再赘述。

7.5 面向对象软件测试用例设计

和传统测试用例设计不同，传统测试是由软件的输入-加工-输出视图或个体模块的算法细节驱动的，面向对象测试关注于设计合适的操作序列以测试类的状态。

Berard 提出了一些测试用例的设计方法，主要原则包括：

- 对每个测试用例应当给予特殊的标识，并且还应当与测试的类有明确的联系。
- 测试目的应当明确。
- 应当为每个测试用例开发一个测试步骤列表。这个列表应包含以下一些内容：

① 列出所要测试的对象的专门说明。

② 列出将要作为测试结果运行的消息和操作。

③ 列出测试对象可能发生的例外情况。

④ 列出外部条件（即为了正确对软件进行测试所必须有的外部环境的变化）。

⑤ 列出为了帮助理解和实现测试所需要的附加信息。

7.5.1 传统测试用例设计方法的可用性

如前所述，尽管 OO 软件的局域性、封装性、信息隐藏、继承性和对象的抽象这些特性使得测试用例设计带来了额外的麻烦和困难，但黑盒测试技术不仅适用于传统软件，也适用于 OO 软件测试，use cases 可以为黑盒及基于状态的测试的设计提供有用的输入。白盒测试也用于 OO 软件类的操作定义，基本路径、循环测试或数据流技术，可以帮助保证已经测试了操作中的每一条语句。但 OO 软件中许多类的操作结构简学明了，所以有人认为在类层上测试可能要比传统软件中的白盒测试方便。

7.5.2 基于故障的测试

在 OO 软件中，基于故障的测试具有较高的发现可能故障的能力。由于系统以满足用户的需求为目的，因此，基于故障的测试要从分析模型开始，考察可能发生的故障。为了确定这些故障是否存在，可设计用例去执行设计或代码。

看一个简单的例子。软件开发人员经常忽略问题的边界，例如，当测试 Divide 操作（该操作对负数和零返回错误）时，我们考虑边界，“零本身”用于检查是否程序员犯了如下错误：

```
if (x>O) divide();
```

而不是正确的：

```
if (x>=O) divide();
```

当然，这种方法的有效性依赖于测试人员如何感觉“似乎可能的故障”，如果 OO 系统中的真实故障被感觉为“难以置信的”，则本方法实质上不比任何随机测试技术好。但是，如果可以从分析和设计模型进行深入的检查，那么基于故障的测试则可以用相当低的工作量来发现大量的错误。

基于故障测试也可以用于集成测试，集成测试可以发现消息联系中“可能的故障”。“可能的故障”一般为意料之外的结果、错误地使用了操作/消息、不正确的引用等。为了确定由操作（功能）引起的可能故障必须检查操作的行为。

这种方法除用于操作测试外，还可用于属性测试，用以确定其对于不同类型的对象行为是否赋予了正确的属性值。因为一个对象的“属性”是由其赋予属性的值定义的。

应当指出，集成测试是从客户对象（主动），而不是从服务器对象（被动）上发现错误。正如传统的软件集成测试是把注意力集中在调用代码，而不是被调用代码一样，即发现客户对象中“可能的故障”。

7.5.3　基于场景的测试

基于故障的测试减少了两种主要类型的错误：

① 不正确的规格说明，如做了用户不需要的功能，也可能缺少了用户需要的功能。

② 没有考虑子系统间的交互作用，如一个子系统（事件或数据流等）的建立，导致其他子系统的失败。

基于场景的测试主要关注用户需要做什么，而不是产品能做什么，即从用户任务（使用用例）中找出用户要做什么及如何去执行。

这种基于场景的测试有助于在一个单元测试情况下检查多重系统。所以基于场景测试用例的测试比基于故障测试的更实际（接近用户），而且更复杂一点。

例如：考察一个 OA 软件的基于场景测试的用例设计。

使用用例：确定最终设计。

背景：打印最终设计，并能从预览窗口上发现一些不易察觉的错误。

其执行事件序列：打印整个文件；对文件进行剪切、复制、粘贴、移动等修改操作；修改文件后，进行文件预览；进行多页预览。

显然，测试人员希望发现预览和编辑这两个软件功能是否能够相互依赖，否则就会产生错误。



7.5.4 OO 类的随机测试

如果一个类有多个操作（功能），这些操作（功能）序列有多种排列。而这种不变化的操作序列可随机产生，用这种可随机排列的序列来检查不同类实例的生存史，就叫随机测试。

例如，一个银行信用卡的应用，其中有一个类：账户（account）。该 account 的操作有 open、setup、deposit、withdraw、balance、summarize、creditlimit 和 close。这些操作中的每一项都可用于计算，但 open、close 必须在其他计算的任何一个操作前后执行，即使 open 和 close 有这种限制，这些操作仍有多种排列，所以一个不同变化的操作序列可由于应用不同而随机产生。如一个 account 实例的最小行为转换期可包括以下操作：

open+setup+deposit+withdraw+close

这表示了对 account 的最小测试序列。然而，在下面序列中可能发生大量的其他行为：
open+setup+deposit+ [deposit|withdraw|balance|summarize|creditLimit] +withdraw+close

由此可以随机产生一系列不同的操作序列，例如，

测试用例 1：open+setup+deposit+deposit+balance+summarize+withdraw+close

测试用例 2：open+setup+deposit+withdraw+deposit+balance+creditLimit+withdraw+close

可以执行这些测试和其他的随机顺序测试，以测试不同的类实例生命历史。

7.5.5 类层次的分割测试

这种测试可以减少用完全相同的方式检查类测试用例的数目。这很像传统软件测试中的等价类划分测试。分割测试又可分为三种。

- 基于状态的分割。按类操作是否改变类的状态来进行分割（归类）。这里仍用 account 类为例，改变状态的操作有 deposit、withdraw，不改变状态的操作有 balance、summarize、creditlimit。如果测试按检查类操作是否改变类状态来设计，则结果如下：

  [用例 1]：执行操作改变状态

  open+setup+deposit+deposit+withdraw+withdraw+close

  [用例 2]：执行操作不改变状态

  open+setup+deposit+summarize+creditlimit+withdraw+close

- 基于属性的分割。按类操作所用到的属性来分割（归类），如果仍以一个 account

类为例，其属性 creditlimit 能被分割为三种操作：用 creditlimit 的操作，修改 creditlimit 的操作，不用也不修改 creditlimit 的操作。这样，测试序列就可按每种分割来设计。

- 基于类型的分割。按完成的功能分割（归类）。例如，在 account 类的操作中，可以分割为初始操作：open、setup；计算操作：deposit、withdraw；查询操作：balance、summarize、creditlimit；终止操作：close。

7.5.6 由行为模型（状态、活动、顺序和合作图）导出的测试

状态转换图（STD）可以用来帮助导出类的动态行为的测试序列，以及这些类与之合作的类的动态行为测试序列。

为了说明问题，仍使用前面讨论过的 account 类。开始由 empty acct 状态转换为 setup acct 状态。类实例的大多数行为发生在 working acct 状态中。而最后，取款和关闭分别使 account 类转换到 non-working acct 和 dead acct 状态。如图 7-3 所示为状态转换图（STD）。

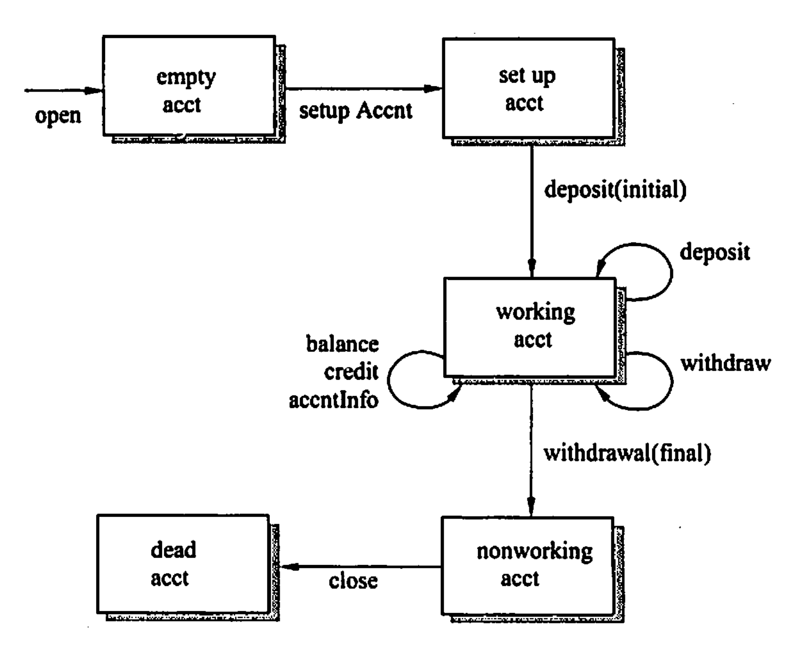


图 7-3　状态转换图（STD）

这样，设计的测试用例应当是完成所有的状态转换。换句话说，操作序列应当能导致 account 类所有允许的状态进行转换。



测试用例 1：open+setupAcct+deposit(initial)+withdraw(final)+close；

应该注意，该序列等同于一个最小测试序列，加入其他测试序列到最小序列中。

测试用例 2：open+setupAccnt+deposit(initial) +deposit+balance+credit+withdraw(final)+close；

测试用例 3：open+setupAccnt+deposit(initial) +deposit+withdraw+accntInfo+withdraw (final) +close。

还可以导出更多的测试用例，以保证该类所有行为被充分检查，在类行为导致与一个或多个类协作的情况下，可使用多个 STD 去跟踪系统的行为流。

面向对象测试的整体目标——以最小的工作量发现最多的错误，和传统软件测试的目标是一致的，但是由于 OO 软件具有的特殊性质，在测试的策略和战术上有很大不同。测试的视角扩大到包括复审分析和设计模型，此外，测试的焦点从过程构件（模块）移向了类。

① OOA（Object-Oriented Analysis）和 OOD（Object-Oriented Design）的评审与传统软件的分析和设计相同，应给出相应的评审检查表。

② OOP（Object-Oriented Programming）后，单元和组装测试策略必须作相应的改变。

③ 测试用例设计必须说明 OO 软件特有的性质。

第 8 章　应用负载压力测试

8.1　负载压力测试概述

8.1.1　负载压力基础概念

系统的负载压力是指系统在某种指定软件、硬件以及网络环境下承受的流量，例如并发用户数、持续运行时间、数据量等，其中并发用户数是负载压力的重要体现。例如当一个应用程序在少量用户同时使用的时候，程序可能会正常运行，然而，当有大量用户同时使用时，可能会出现功能失效、性能衰减，甚至系统崩溃的现象。

8.1.2　负载压力测试基础概念

负载压力测试是指在一定约束条件下测试系统所能承受的并发用户量、运行时间、数据量，以确定系统所能承受的最大负载压力。

负载压力测试有助于确认被测系统是否能够支持性能需求，以及预期的负载增长等。负载压力测试不只是关注不同负载场景下的响应时间等指标，它也要通过测试来发现在不同负载场景下会出现的，例如速度变慢、内存泄漏等问题的原因。因此，应该在开发过程中尽可能早地进行负载压力测试。

负载压力测试是性能测试的重要组成部分，负载压力测试包括并发性能测试、疲劳强度测试、大数据量测试等内容。下面分别介绍这些概念。

1．性能测试

系统的性能是一个很大的概念，覆盖面非常广泛，对一个软件系统而言，包括执行效率、资源占用、稳定性、安全性、兼容性、可扩展性、可靠性等，我们这里重点讨论的负载压力是系统性能的一个重要方面。性能测试用来保证产品发布后系统的性能能够满足用户需求。性能测试在软件质量保证中起重要作用。通常情况下存在性能调优与性能评测两种性能测试策略。

2．性能评测

性能评测主要内容包括以下两项内容。

- 在真实环境下，检查系统服务等级的满足情况，评估并报告整个系统的性能。
- 对系统的未来容量作出预测和规划。

性能评测是性能调优的基础。

3．性能调优

性能调优的步骤如下。

- 查找形成系统瓶颈或者故障的根本原因。
- 进行性能调整和优化。
- 评估性能调整的效果。

在通常情况下，性能调优的过程是上述步骤循环执行的过程，以实现目标。

4．负载测试

负载测试是通过逐步增加系统负载，测试系统性能的变化，并最终确定在满足性能指标的情况下，系统所能承受的最大负载量的测试。

5．压力测试

压力测试是通过逐步增加系统负载，测试系统性能的变化，并最终确定在什么负载条件下系统性能处于失效状态，并以此来获得系统能提供的最大服务级别的测试。通俗地讲，压力测试是为了发现在什么条件下系统的性能会变得不可接受。

可见，压力测试是一种特定类型的负载测试。例如，访问一个页面的响应时间规定为不超过 1 秒，负载测试就是测试在响应时间为 1 秒时，系统所能承受的最大并发访问用户的数量，而压力测试就是测试系统在多大的并发访问用户数量下，响应时间不可接受，例如超过 1 分钟（定义为失效状态）。

6．并发性能测试

并发性能测试的过程，是一个负载测试和压力测试的过程。即逐渐增加并发用户数负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标、资源监控指标等来确定系统并发性能的过程。并发性能测试是负载压力测试中的重要内容。

从一个完整解决方案的角度考虑，并发性能测试概括为以下 3 类。

- 应用在客户端性能的测试；
- 应用在网络上性能的测试；
- 应用在服务器上性能的测试。

7．疲劳强度测试

通常是采用系统稳定运行情况下能够支持的最大并发用户数，或者日常运行用户数，持续执行一段时间业务，保证达到系统疲劳强度需求的业务量，通过综合分析交易执行指标和资源监控指标，来确定系统处理最大工作量强度性能的过程。一般情况下利用疲劳强度测试来模拟系统日常业务操作。

在后面章节的实例部分有相关举例。

8．大数据量测试

大数据量测试包括独立的数据量测试和综合数据量测试两类。

独立的数据量测试指针对某些系统存储、传输、统计、查询等业务进行的大数据量测试。

综合数据量测试指和压力性能测试、负载性能测试、疲劳性能测试相结合的综合测试。

8.1.3　负载压力测试目的

这是一个很重要的问题，也是测试前首先要考虑的问题。

我们经常听到"很多人都在使用系统时，响应时间太慢了，到底问题在哪里"这样的用户抱怨。类似的问题还有"要花多少时间做完一笔交易；什么样的配置提供了最好的性能；系统能在无错情况下承担多大及多长时间的负载；这些升级对系统性能影响多大；服务器应该选择哪些硬件与软件；在没有较大性能衰减的前提下，系统能够承受多大负载；哪些因素降低交易响应时间"等等，这样直观的问题描述代表了测试需求，也由此决定了测试目的。

负载压力测试的目的可以概括为以下几个方面。

- 在真实环境下检测系统性能，评估系统性能以及服务等级的满足情况。

例如电信计费软件，众所周知，每月 20 日左右是市话交费的高峰期，全市几千个收费网点同时启动。收费过程一般分为两步，首先要根据用户提出的电话号码来查询出其当月产生费用，然后收取现金并将此用户修改为已交费状态。一个看起来简单的两个步骤，当成百上千的终端同时执行这样的操作时情况就大不一样了，如此众多的交易同时发生，对应用程序本身、操作系统、中心数据库服务器、中间件服务器、网络设备的承受力都是一个严峻的考验。决策者需要模拟系统负载压力，预见软件的并发承受力，这是在测试阶段就应该解决的重要问题。

一个企业自己组织力量或委托软件公司代为开发的应用系统，在生产环境中实际使用起来以后，往往会产生这样一个问题，即这套系统能不能承受大量的并发用户同时访问，这个问题是系统负载压力需求的体现。

这里强调在真实环境下检测系统性能，在实施过程中大家认为这样做会遇到很多阻力，比如系统上线运行之后，真实环境下不允许负载压力测试为系统带来大量的垃圾数据，测试数据与真实业务数据混在一起无法控制测试结果，负载压力测试如果使服务器宕机会给系统带来巨大损失等。那么在这种条件不允许的情况下，应该采用什么样的措施弥补呢？我们可以使用一种"模拟环境"来做测试，这种环境是指与实际真实应用环境基本等级保持一致的测试环境。



- 预见系统负载压力承受力，在应用实际部署之前，评估系统性能。

目前的大多数公司企业需要支持成百上千名用户，各类应用环境，以及由不同供应商的元件组装起来的复杂产品。难以预知的用户负载和越来越复杂的应用程序，使公司时时担忧会发生投放性能差，用户遭受反应慢，系统失灵等问题。其结果就是导致公司收益的损失。

检测系统性能强调对系统当前性能的评估，通过评估，可以在应用实际部署之前，预见系统负载压力承受力。这种测试的意义在于指导系统总体设计，既可以避免浪费不必要的人力、物力和财力，又避免硬件和软件的设计不匹配，使系统具有更长、更健壮的生命力。

如何确定系统的“负载压力承受力”是一个非常复杂且关键的问题，我们会在“负载压力测试需求分析”一节中详细论述。

对于系统性能检测，有时我们所从事的工作仅仅是被动监控一些性能指标，而预见系统负载压力承受力，则不可避免地会借助自动化的负载压力测试工具。

- 分析系统瓶颈、优化系统。

系统性能检测和预见为分析系统瓶颈和优化提供了原始数据，打好了基础。

瓶颈这个术语来源于玻璃瓶与瓶身相比收缩了的部分。收缩的瓶颈将引起流量的下降，从而限制了液体流出瓶外的速度。类似的，在负载压力测试中，“瓶颈”这个术语用来描述那些限制系统负载压力性能的因素。我们给系统瓶颈一个简单定义，即应用系统中导致系统性能大幅下降的原因。瓶颈大大降低了系统性能，测试工程师的职责之一，就是降低或者消除系统中的瓶颈。一般情况下，发现瓶颈并找出原因并不是件容易的事。很多时候，你可能无法准确定位系统瓶颈之所在。瓶颈可能定位在硬件中，也可能定位在软件中。对软件来讲，可能定位在开发的应用程序中，也可能定位在操作系统或者数据库内部，对于后者，我们是无能为力的。数据库和操作系统的开发者们都一直在测试其产品的新版本，以期能尽其所能地排除产品中存在的所有瓶颈。硬件中的瓶颈可能会非常容易排除，一般来讲，解决硬件瓶颈的方法只是简单地向系统中添加 CPU、磁盘或者内存等，如果硬件瓶颈是由于系统缓冲区设计或内存总线造成的，那么通常情况下我们就无能为力了。硬件瓶颈与软件瓶颈相比，我们更建议先解决软件瓶颈，原因有三，其一是软件瓶颈往往导致系统性能衰减更快，反过来讲，消除软件瓶颈，系统性能提升更快；其二是人为因素更易导致软件瓶颈，要消除软件瓶颈，开发人员会更主动，并且可以节省资源；其三，盲目增加硬件则无形中增加维护费用，将来，软硬件不匹配的问题终究还会暴露出来。

优化调整系统是在发现瓶颈，故障定位之后要完成的事情，实现优化之后即可消除瓶颈，提高性能。我们建议将负载压力性能问题分为两类：一类是需要优化的性能问题，

这类问题可能导致系统性能大幅度下降，或者给系统造成破坏，也可能性能不会下降许多；另一类是非系统优化所能解决的性能问题。我们这里讨论的是前者，导致系统性能下降的因素来自许多方面，例如 I/O 过载、内存不足、数据库资源匮乏、网络速度低、硬件资源不足、操作系统资源不足、应用程序架构存在缺陷，软硬件配置不恰当等等。优化调整即是对症下药，做到药到病除。我们来看一个例子，如果是磁盘 I/O 导致了系统瓶颈，那么消除它的方法可能是重新设计数据库或者提高系统能力。

在系统优化调整领域，有许多指导性文件，例如 IBM 针对其产品 DB2、WebSphere、CM、Portal 等都提供了专门的 Tuning Guide，并且配合以培训。我们知道，各个组成部分的最优并不代表系统性能可以达到最优，每个组成单元都具备一些调优指标，每个指标的调整并非最大就好，或者最小就好，也很少存在在某个范围就最好这种情况。我们给测试工程师的建议是，调优的最终目的是各个指标的调整取得系统的平衡点，也即达到了系统性能的最佳点。讲到这里，我们会想到吴清源大师的“六合之棋”，他曾经提出围棋的目标不是局限于边角，而是应该很好地保持全体的平衡，站在一个很高的角度去看待。

由此可见，负载压力测试将为企业项目的实施提供信心，帮助用户正确地进行容量规划，实现软硬件投资合理化，最终交付高质量的系统，避免项目投产失败，保证用户的投资得到相应的回报。

8.1.4 负载压力测试策略

负载压力测试可以采取利用手工进行测试和利用自动化负载压力测试工具进行测试两种测试策略。

大多数工程师掌握手工测试技巧，比如，可以手工模拟负载压力，方法是找若干台电脑和同样数目的操作人员，在同一时刻进行操作，然后用秒表记录下响应时间，这样的手工测试方法可以大致反映系统所能承受的负载压力情况。但是，这种方法需要大量的人员和机器设备，而且测试人员的同步问题无法解决，更无法捕捉程序内部的变化情况。利用自动化负载压力测试工具进行测试可以很好地解决这些问题。利用自动化负载压力测试工具可以在一台或几台 PC 机上，模拟成百或上千的虚拟用户同时执行业务的情景，通过可重复的、真实的测试能够彻底地度量应用的性能，确定问题所在。

可见，负载压力测试的发展趋势是，利用自动化的测试工具进行测试，当然在没有工具的情况下，我们也可以通过手工测试对系统承受负载压力情况做一个近似的评估。下面重点介绍一下利用自动化测试工具进行负载压力测试的策略，分别是利用商业化测试工具进行测试、利用开放资源测试工具进行测试和自主开发工具进行测试。



- 利用商业化测试工具进行测试。

利用商业化的自动化测试工具是进行负载压力测试的主要手段，知名的商业化的测试工具，比如 LoadRunner、QALoad 等，适用范围非常广，一般都经过了长时间的市场检验，测试效果得到业界的普遍认可，测试结果具有一定的可比性，并且厂商一般都能提供很好的技术支持，其版本的升级也会得到保证。但是商业化的自动化测试工具一般价格较高，如果考虑价格因素，那么利用开放资源工具进行测试也是一个不错的策略。

- 开放资源测试工具进行测试。

开放资源被定义为用户不侵犯任何专利权和著作权，以及无需通过专利使用权转让，就可以获取、检测、更改的软件源代码，这意味着任何人都有权访问、修改、改进或重新分配源代码。开放资源的理念是，当人们在已存在的工具上共同开发时，最终产品会更加先进。简而言之，很多企业和个体都会从中获益。开放资源的最大优点是测试工具是免费的。

现在我们来看看开放资源性能测试工具。

最流行的几个开放资源性能测试工具是：开放系统测试体系 OpenSTA、TestMaker 和 JMeter。这些工具中的每一个都能提供完成负载压力测试所需的功能，现存的多种开放资源测试工具都是可获得的。下面列举几个例子。

① 开放系统测试体系——OpenSTA（http://portal.opensta.org/）。

OpenSTA 是 Windows 平台、分布式的软件测试体系，基于 CORBA（Common Object Request Broker Architecture）。OpenSTA 能产生数百或数千个虚拟用户，最初用于测试基于 Web 的应用软件。此工具还为用户响应时间和平台应用软件（包括应用服务器、数据库服务器、Web 服务器）的资源占用信息监控提供了图形化标准。

OpenSTA 具有一种简单的脚本语言，即脚本控制语言（SCL）。SCL 与商业性能测试工具一样，使用户能够创建测试脚本。它能将输入数据参数化，从外部文件读入参数数据。如图 8-1 所示是 OpenSTA 的脚本模型界面。

② TestMaker（http://www.pushtotest.com/）。

它是一种基于 Java 的架构，能够创建测试代理，以用于衡量应用软件和 Web 服务器的性能。TestMaker 可以在 Windows、Linux 和 UNIX 平台上运行，可以用它创建针对 Web 应用的测试案例，而不管这些应用是基于 J2EE 平台还是.NET 平台。TestMaker 支持各种不同的协议，例如 HTTP/HTTPS、TCP/IP、SOAP 以及 XML。TestMaker 的脚本语言是一种开放资源语言，叫做 Jython。Jython 其实是 Python 语言的 Java 实现形式。Jython 除了给开发者提供所有的 Java 对象外，还提供 Python 的面向对象的环境。TestMaker 包含一个代理日志，同商品化测试工具所提供的功能类似。

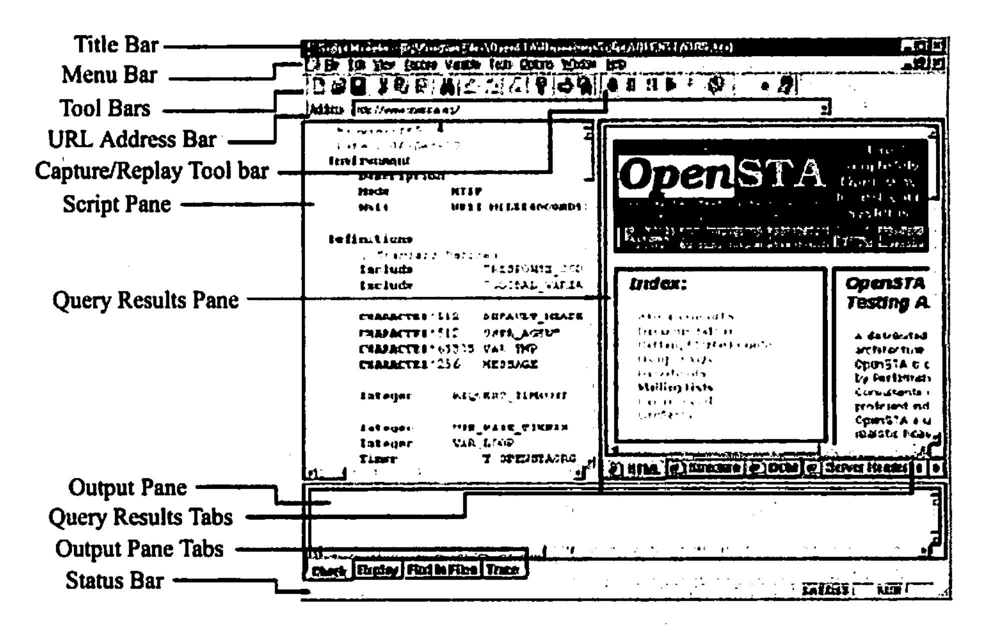


图 8-1 OpenSTA 脚本模型界面

③ Apache JMeter（http://jakarta.apache.org/jmeter/）。

Apache JMeter 是一种纯粹的 Java 应用软件，用于测试功能和衡量性能。JMeter 最初是基于 Apache Tomcat 设计的，用于测试 Web 应用软件的性能，但是目前，开放资源发展联盟将此产品的应用扩展得更广泛了，Apache JMeter 同时用于功能测试和负载压力测试。应用这一软件可以测试 Java 对象、JDBC、数据库、Perl 脚本、Web 服务器和应用服务器等。

和商品化测试工具一样，Apache JMeter 的代理记录可以记录浏览器和 Web 服务器之间的通信。并且，由于 JMeter 是 100%Java 的，所以它不受平台约束。

④ 自主开发工具测试。

自主开发测试即开发自己的负载压力测试程序或者工具。

例如，一个简单的 Web 应用测试工具可以这样构建。首先编写一个对每一个模拟客户机运行一个线程的程序。每一个线程需要与服务器通信，可能使用 Java、Net、URL 类。这种方法能够达到基本的 HTTP 客户机模拟，它可以执行 GET 和 PUT。每个线程需要做的就是发送 HTTP 请求，收集回复。这一组行动可以相当容易地抽象到一个单独的配置文件中。很快地，就得到一个基本的负载测试工具。同时可能需要增加一些配置选项以确定运行多少个线程（模拟的客户机），以及它们是同时开始，还是慢慢增加负载。当然，需要对与服务器的交互计时，因为这是要测试的核心内容，响应时间。



后面章节中，我们会详细论述开发负载压力测试程序或者工具的一些思路。

8.1.5 产品生命周期中负载压力测试计划

我们知道，在项目的不同阶段都需要进行负载压力性能测试，而测试是需要必要的资源的，所以应该为此制定相应的计划。这里提供了 5 点计划安排，它们能确保系统负载压力性能满足需求。

（1）在需求分析中充分关注负载压力性能

在需求分析阶段，主要的焦点是为系统中共享的和有限的资源进行需求分析。例如，一个网络联接既是共享的又是有限的资源；一个数据库表是一个共享的资源；线程是一个有限的资源。如果没有正确的设计，这些在以后的各阶段将引发严重问题。

为了突出负载压力性能需求分析，有时需要为负载压力性能分析分配大约 10%的时间，不同的设计选择对于负载压力性能的影响是不同的。测试工程师需要掌握负载压力性能目标设计方法，同时应该具备与确定负载压力性能需求相关的体系结构资料，需求分析应该与体系结构分析结合进行。

（2）从设计中得到负载压力性能指标

设计者应当清楚地了解不同设计对负载压力性能的影响，在设计的各个方面应该充分考虑负载压力性能设计各方的意见，给出负载压力性能的预期指标。

如果设计中系统应用了第三方产品，例如，中间件或者数据库产品，则应要求第三方产品提供商能够对其产品进行性能验证和设计，识别与其产品有关的负载压力性能问题。

为了突出负载压力性能的重要性，在预算方面也应当留出专门的资金，如为负载压力性能方面分配 10%的资金预算是一个安全的选择。

设计中还应该考虑应用规模和数据量的可升级性。应用分布的规模可能依赖于分布组件的需求级别、事务处理机制和模式等，数据量的升级将要求设计中包含专门处理大数据集处理的内容。

（3）开发阶段创建一个负载压力性能测试环境

开发阶段开始时的负载压力性能任务是建立负载压力性能测试环境。需要进行以下工作：

- 确保合理精确的测试环境，并且此环境可重用；
- 为测试环境制定规则的负载压力性能测试时间表，如果测试环境是共享的，负载压力性能测试不能与其他活动同时发生；
- 选择一个负载压力性能测试工具。

（4）验收阶段在多等级范围内测试并调优

测试要如实表现系统的主要应用，测试系统的可升级性，例如，可确定共享资源如

何响应增长的负载，也可确定受限资源在哪个阶段开始用尽或者成为瓶颈。

验收测试结果可以统计负载压力性能、比较负载压力性能，以及报告异常，提供分析依据。

（5）运行阶段持续监控系统负载压力性能

监控系统在正常运行状态下的负载压力性能，识别系统性能的倾向，确定何种条件下负载压力性能超过可接受范围等。

8.1.6　负载压力测试中的盲点

就像汽车的观后镜存在“盲点”一样，许多负载压力测试工具也存在“盲点”，会给使用工具的测试工程师一种盲目的安全感。这个盲点是什么呢？就是在负载压力测试中，不进行功能校验，当功能错误发生时，测试工具不能够记录产生的错误，这就忽略了负载压力情况下的功能不稳定问题。目前分布式应用部署结构、Web 技术的发展等极大地扩大了这些“盲点”发生的可能，忽视这些“盲点”，必然会导致结果的不可信，甚至导致结果有害。

所以负载压力测试期间必须要进行必要的功能内容校验，换句话讲，没有正确的功能保证，负载压力性能测试就失去了意义。那么，如何做功能内容校验呢？我们认为只有在负载压力测试过程中记录所有虚拟用户的操作，及服务器的响应，才有助于判断功能错误，这就是当前负载压力测试技术发展的最大挑战。测试过程中的附加记录会导致资源消耗、操作行为增加以及产生大量日志等问题。

目前有些负载压力测试工具能够将负载压力测试与功能校验融为一体，尽量减少此“盲点”。如果使用的工具不具备此功能，那么就必须借助于一些辅助手段来克服“盲点”，视而不见肯定不是一个明智的选择。

8.2　负载压力测试解决方案

系统的负载压力主要包括并发负载、疲劳强度以及大数据量等，前面我们已经讨论过这些负载压力测试的概念，这里我们讨论具体测试项的解决方案。

8.2.1　并发性能测试

系统的并发性能是负载压力性能的最主要的组成部分，首先我们来讨论什么是“并发”。对一个系统来讲，某些业务操作对特定角色用户来讲存在很大的同时操作的可能性。例如，网上购物系统的订单提交、订票系统的票源查询、人力资源系统月末及年末报表上传等，客户端大量的并发操作提高了网络的吞吐量，加剧了服务器资源互斥访问

冲突，加大了数据库死锁的可能。这样的负载压力轻则会导致系统性能低下，重则会对系统造成破坏，给用户带来经济损失。因此并发性能的测试对于保证系统的性能是非常关键的。

那么，在实际操作中，我们采取什么样的方案来实施这一类操作呢？首先并发负载压力的实施是在客户端，负载压力的传输介质是网络，最终压力会到达后台各类服务器，包括 Web 服务器、应用服务器、数据库服务器以及系统必须的服务器，例如认证服务器、检索服务器等。所以，在并发性能测试过程中，关注点包括客户端的性能、应用在网络上的性能以及应用在服务器上的性能。为什么这些内容都是必须的呢？大家知道，测试是要定位问题，目的是为了解决问题，这些关注点正是定位问题的必要条件。下面将详细论述这些重点内容。

1．应用在客户端性能的测试

在客户端模拟大量并发用户执行不同业务操作，达到实施负载压力的目的。

采用负载压力测试工具来模拟大量并发用户，模拟机制如图 8-2 所示，主要组成部分包括主控台、代理机以及被测服务器，各部分采用网络连接。主控台负责管理各个代理以及收集各代理测试数据，代理负责模拟虚拟用户加压。在每次并发性能测试中，只有一台主控台，但可以有多个代理。

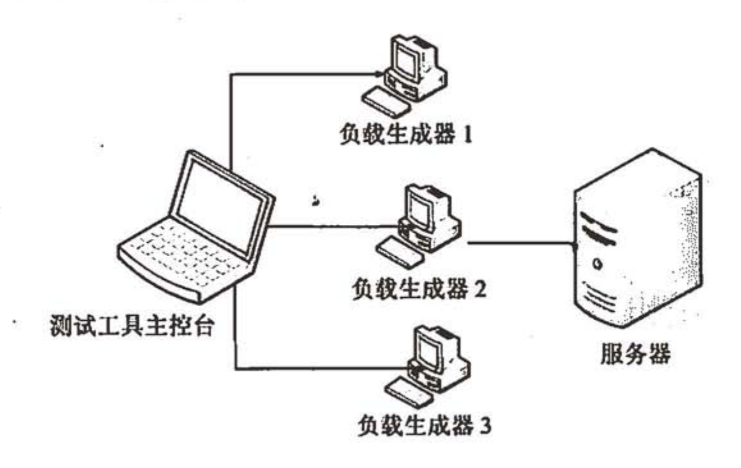


图 8-2 模拟机制

如何模拟负载压力呢？总的原则是最大限度地模拟真实负载压力。要做到这一点，既是对测试工具的考验，又是对测试工程师经验与智慧的考验。

测试工具怎样模拟真实的负载压力呢？以 LoadRunner 负载压力测试工具为例，看看工具是如何实现的。

要模拟真实的负载压力做测试，必须创建方案，方案是用以模拟现实生活中用户的方式。方案包含有关如何模拟实际用户的信息：虚拟用户（Vuser）组、Vuser 将运行的测试脚本，以及用于运行脚本的负载生成器计算机。

如果选择创建常规手动方案，则会将选择的每个脚本分配给 Vuser 组。然后，可以为每个 Vuser 组分配多个 Vuser。可以指示某个组中的所有 Vuser 在同一台负载生成器计算机上运行相同的脚本，也可以为组中的各个 Vuser 分配不同的脚本和负载生成器。使用百分比模式创建手动方案，可以定义方案中要使用的 Vuser 总数，并为每个脚本分配负载生成器和占总数一定百分比的 Vuser。

可以设计面向目标的方案，在面向目标的方案中，可以定义所希望实现的测试目标，LoadRunner 将根据定义的目标自动创建一个方案。在一个面向目标的方案中，可以定义五种类型的目标：虚拟用户数、每秒点击次数（仅 Web Vuser）、每秒事务数、每分钟页面数（仅 Web Vuser）或事务响应时间。要定义每秒事务数或事务响应时间目标类型，脚本中必须包含事务。对于每种目标类型，可以定义脚本中希望测试的事务。下面分别来分析五种类型的目标。

- 虚拟用户目标类型：测试应用程序可以同时运行多少个 Vuser。
- 每秒点击次数、每分钟页面数或每秒事务数：测试服务器的稳定性。需要指定 LoadRunner 运行的 Vuser 范围（最大值、最小值），以及每秒事务数目标类型的“事务名称”。Controller（测试工具的主控台）将尽量使用最少数量的 Vuser 来达到定义的目标。如果使用最小 Vuser 数不能达到该目标，则 Controller 将逐渐增加 Vuser 数，直到达到所定义的最大数。如果使用指定的最大 Vuser 数仍不能达到指定的目标，Controller 将增加 Vuser 数，并再次执行方案。
- 事务响应时间目标类型：测试在期望的事务响应时间内可以同时运行多少个 Vuser，在脚本中指定想要测试的事务的名称以及 LoadRunner 要运行的 Vuser 数量范围（最大值、最小值）。指定的“事务响应时间”应该是一个预定义的阈值。例如，如果希望用户在 5 秒钟之内登录到某个电子商务站点，请将可接受的最长事务响应时间指定为 5 秒。将最大和最小 Vuser 数设置为希望能够同时提供服务的最大和最小用户数。如果方案没有达到定义的最大事务响应时间，则服务器能够在合理的时间间隔内，对想要同时提供服务的指定数量的用户作出响应。如果在仅执行部分 Vuser 后就达到定义的响应时间，或如果接收到消息，提示如果 Controller 使用定义的最大 Vuser 数，响应时间将超出指定值，那么，应该考虑修补应用程序和/或升级服务器的软硬件。

下面来谈谈方案计划。方案的主要内容是确定如何开展测试，以准确描绘用户行为（操作类型和这些操作的计时等，由 Vuser 脚本表示）。可以在一段延迟之后开始执行方



案。可以指定 LoadRunner 自发出 Run 命令以来等待的分钟数，也可以指定让方案开始的特定时间。使用计划生成器，可以对手动方案进行计时设置，从而限制方案的执行持续时间，或 Vuser 组在方案中的持续时间。通过指定方案或 Vuser 组应处于“正在运行”状态的分钟数，可以限制执行持续时间。方案或组到达其时间限制时就结束。对于手动方案，还可以规定在某一时间段内 LoadRunner 启动和停止的 Vuser 的数量。在指定的时间量内，可以指定 LoadRunner 应同时启动/停止，Vuser 组中所有的 Vuser，还是仅启动/停止一定数量的 Vuser。需要注意的是，Vuser 脚本中的集合点将干扰已计划好的方案。如果脚本包含集合点，则方案将不会按计划运行。

在方案运行期间，可以通过使用集合点指示多个 Vuser 同时执行任务。集合点可以在服务器上创建密集的用户负载，并使 LoadRunner 能够测量服务器在负载状态下的性能。假设有 10 个 Vuser 同时检查账户信息时，需要估量某个基于 Web 的银行系统如何执行操作。为了模拟服务器上要求的用户负载，可以指示所有的 Vuser 完全在同一时刻检查账户信息。通过创建集合点，可以确保多个 Vuser 同步操作。当 Vuser 到达某个集合点时，它就会被 Controller 滞留在该处。当达到要求的 Vuser 数或者经过一段指定的时间后，Controller 就会从集合中释放 Vuser。

通过使用 Controller，可以根据如下选择来影响服务器的负载级别：

- 选择在方案运行过程中活动的集合点；
- 选择加入每个集合的 Vuser 数。

例如，要测试银行服务器，可以创建一个包含两个集合点的方案。第一个集合可以确保 1000 个 Vuser 能同时存入现金。第二个集合可以确保另外 1000 个 Vuser 能同时提取现金。如果需要在只有 500 个 Vuser 存入现金的情况下度量服务器的性能，可以停用（禁用）“提取”集合，并指示仅让 500 个 Vuser 参加“存入”集合。下面的过程概述了如何控制服务器上的负载峰值。

- 创建 Vuser 脚本，插入必需的集合点。
- 创建方案。
- 向方案中添加 Vuser 组时，LoadRunner 扫描与该组相关的脚本，在其中搜索集合点的名称，并将这些名称添加到“集合信息”对话框中的列表里。如果创建另外一个运行相同脚本的 Vuser 组，Controller 会将该新的 Vuser 添加到集合中，并更新列表。
- 设置模拟用户负载的级别。
- 通过选择将加入到方案中的集合点，以及加入每个集合的 Vuser 数，可以确定负载的精确级别。
- 设置集合的属性。

- 对于每一个集合，都可以设置集合策略。
- 运行方案。

在运行方案之前，可以同时配置方案的负载生成器和 Vuser 行为。虽然默认设置与大多数环境对应，但是 LoadRunner 允许修改这些设置，以便自定义方案行为。这些设置适用于所有未来的方案运行，并且通常只需设置一次。如果全局方案设置与单个负载生成器的设置不同，则负载生成器设置将替代全局方案。

这里讨论的设置与 Vuser 运行时的设置无关。这些适用于单个 Vuser 或脚本的设置，包含有关日志记录、思考时间、网络、迭代数和浏览器的信息。LoadRunner 导出模式允许配置 LoadRunner 代理程序和其他 LoadRunner 组件的其他设置，称为“专家模式”。

我们使用大量的篇幅论述了测试工具，其实在测试过程中，占主导地位的仍然是测试工程师，测试工程师对测试工具掌握的熟练程度，负载压力测试积累的经验，对新技术的敏感程度，和开发人员及用户的充分交流，学无止境的工作态度都是非常重要的。

测试工程师只有程序设计和开发工具的知识是不够的，必须要懂得系统运转的机理。要具备应用平台、软件架构、数据库系统以及网络环境等方面的知识，这样才能做到尽量分析错误和定位错误。

例如，当对关键功能点录制脚本时，不但要掌握功能点完成的任务，还要熟悉功能的业务运行模式，这就需要测试工程师有业务操作背景。

客户端并发性能测试要得到哪些指标的值呢，以及怎样分析结果值，我们在后面章节会有详细举例论述。

2．应用在网络上性能的测试

这部分主要包括两部分内容，一是应用网络故障分析；二是网络应用性能监控。

应用网络故障分析的测试目标是显示网络带宽、延迟、负载和 TCP 端口的变化是如何影响用户的响应时间的。通过测试，我们可以做到下面几点。

① 优化性能；

② 预测系统响应时间；

③ 确定网络带宽需求；

④ 定位应用程序和网络故障。

借助于网络故障分析工具，可以解决下列问题。

① 使应用跨越多个网段的活动过程变得清晰；

② 提供有关应用效率的统计数据；

③ 模拟最终用户在不同网络配置环境下的响应时间，决定应用投产的网络环境。

网络故障分析工具的工作机理，可以总结为“多个捕捉点，一个分析”。捕捉点即“Agent”，利用主控台“Agent Manager”进行分析，Agent 被动监听数据包来实现实时数

据采集，Agent Manager 完成对所跟踪到的数据的分析，可以自由地将捕捉代理放在不同的平台，例如，Windows 或 UNIX。如图 8-3 所示的主控台为 Management Console，捕捉点为代理探针 Probe。

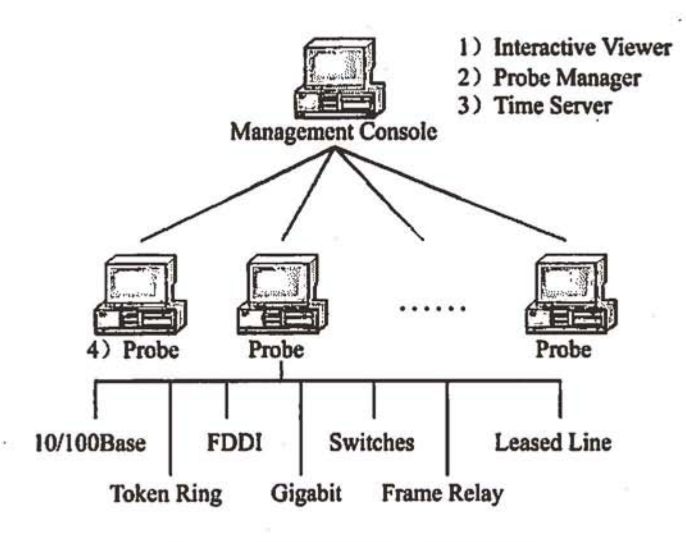


图 8-3 网络故障分析工具工作机理

如图 8-4 所示是一个典型的 Web 架构应用部署图，我们可以在应用逻辑路径上进行多点数据采集，并且在任何两个节点间进行数据整合，测量分段的响应时间，分析应用故障。

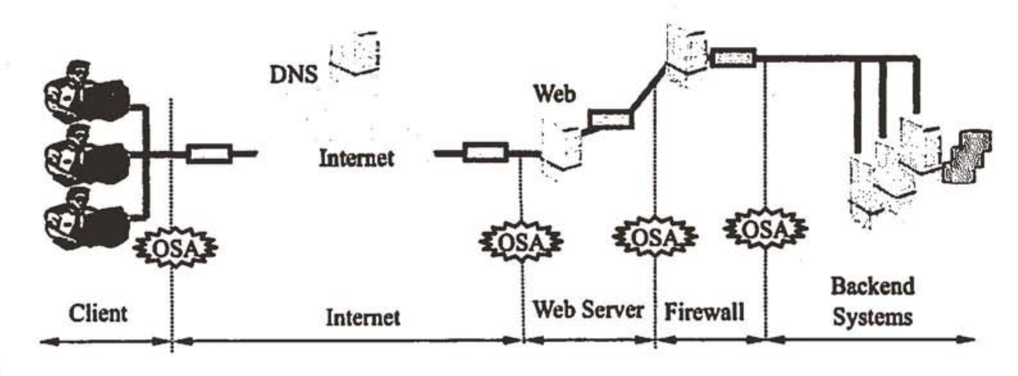
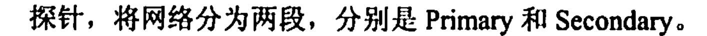


图 8-4 Web 架构应用部署图

有哪些信息可以辅助我们做网络故障分析呢？

- 监控不同探针之间的连接状态，传输的字节数以及通信往返行程次数，如图 8-5 所示，206.40.55.195 为浏览器，www.optinal.com 为 Web服务器，其上各有一个



探针，将网络分为两段，分别是 Primary 和 Secondary。

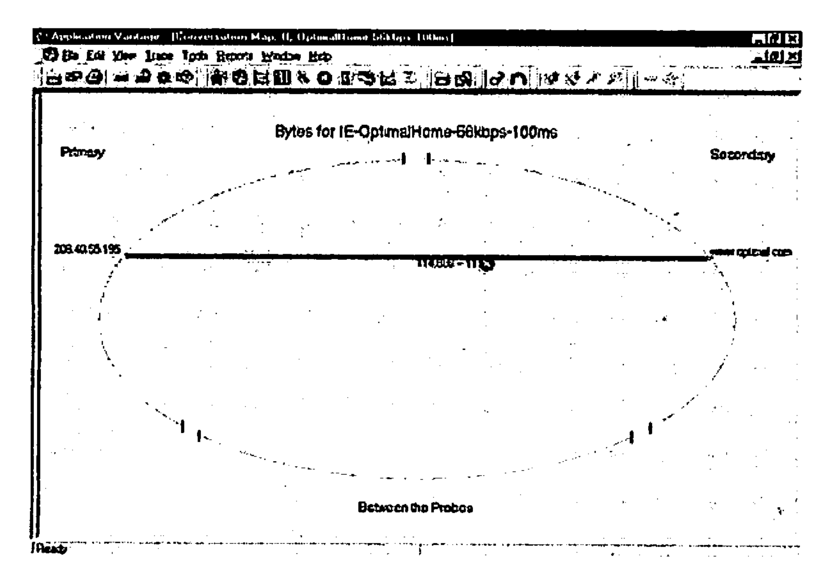


图 8-5　探针之间的连接

- 会话性能概要，监控哪段网络延迟大，带宽对网络双向性能的影响，节点用于处理和用于传输的时间等，如图 8-6 所示。

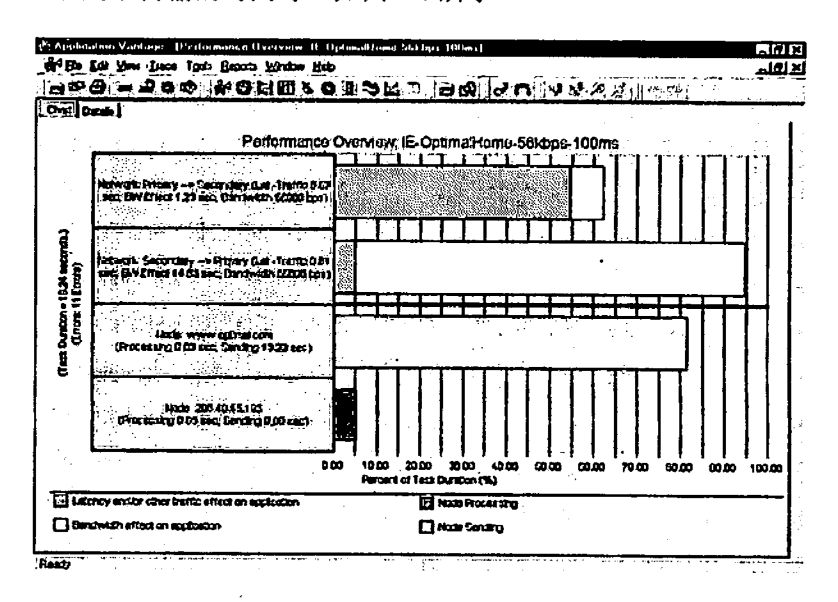


图 8-6　会话性能概要

- 服务器与客户端之间帧传输情况统计，可以监控到与应用相关的帧的分布，对每一个帧可以与相关的数据包关联，并且可以对帧解码，如图 8-7 所示。

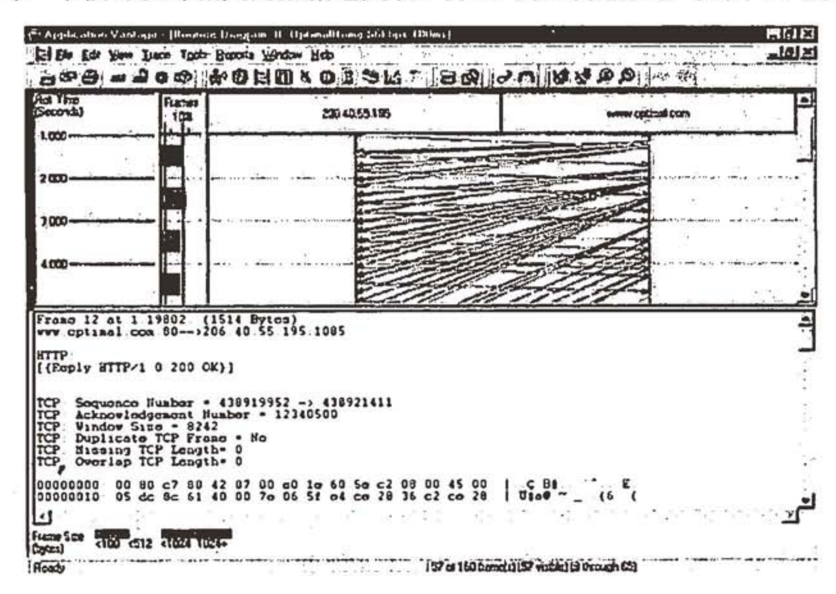


图 8-7 服务器与客户端之间帧传输情况统计

- 服务器与客户端之间传送包信息统计，监控包的详细信息，并且可以将包与帧及线程相关联，如图 8-8 所示。

	Error	Seq	Direction	Sent (sec)	Received (sec)	Travel Time	Bytes	Payload Bytes	Source Node	Source Port	Destination Node	Destination Port	
1		1	->	0.00000	0.26660	0.20008	405	351	206 40 55 19	1084	www optimal	80	HTTP [(GET /H
2		2	<-	0.26671	0.47000	0.21209	1,514	1,460	www optimal	80	206 40 55 19	1084	HTTP [(Reply H
3		3	<-	0.26683	0.69531	0.42048	1,514	1,460	www optimal	80	206 40 55 19	1084	HTTP [(Reply C
4		4	<-	0.26695	0.78284	0.51538	620	566	www optimal	00	206 40 55 19	1084	HTTP [(Reply C
5		5		0.63721	0.63721		60	46	00E01EC05E	0	00E01EC05E	0	LOOPBACK
6		6	->	0.64906	0.86754	0.21049	60	0	206 40 55 19	1084	www optimal	80	TCP Ack Seq# 1
7		7	->	0.70381	0.99773	0.21392	60	0	206 40 55 19	1084	www optimal	00	TCP Ack Seq# 1
8		0	->	0.93763	1.18000	0.25030	322	268	206 40 58 19	1085	www optimal	00	HTTP [(GET /H
9		9	->	0.95607	1.23000	0.20200	322	268	206 40 55 19	1026	www optimal	80	HTTP [(GET /H
10		10	->	0.97033	1.27813	0.30715	322	268	206 40 55 19	1034	www optimal	00	HTTP [(GET /H
11		11	->	1.00301	1.20034	0.20533	60	0	206 40 58 19	1007	www optimal	00	TCP Connection
12		12	<-	1.19802	1.41601	0.21000	1,514	1,460	www optimal	80	206 40 55 19	1008	HTTP [(Reply H
13		13	<-	1.19814	1.48720	0.28914	500	446	www optimal	80	206 40 55 19	1005	HTTP [(Reply C
14		14	<-	1.23010	1.71543	0.47733	1,514	1,460	www optimal	80	206 40 55 19	1086	HTTP [(Reply H
15		15	<-	1.23022	1.84032	0.60210	874	820	www optimal	00	206 40 55 19	1006	HTTP [(Reply C
16		16	<-	1.20015	2.06790	0.77975	1,514	1,460	www optimal	80	206 40 58 19	1084	HTTP [(Reply H
17		17	<-	1.20027	2.10610	0.09704	829	775	www optimal	80	206 40 55 19	1084	HTTP [(Reply C
18		10	<-	1.20034	2.19324	0.90490	60	0	www optimal	80	206 40 55 19	1007	TCP Connection
19		19	->	1.48836	1.70075	0.22033	60	0	206 40 58 19	1085	www optimal	80	TCP Ack Seq# 1
20		20	->	1.57049	1.82893	0.25044	324	270	206 40 55 19	1085	www optimal	00	HTTP [(GET /H
21		21	<-	1.02895	2.41064	0.50169	1,514	1,460	www optimal	80	206 40 55 19	1085	HTTP [(Reply H
22		22	<-	1.82907	2.62776	0.79069	1,514	1,460	www optimal	80	206 40 56 19	1085	HTTP [(Reply C
23		23	<-	1.02919	2.83381	1.00462	1,435	1,381	www optimal	00	206 40 55 19	1085	HTTP [(Reply C
24		24	->	1.04155	2.05926	0.21770	60	0	206 40 58 19	1086	www optimal	80	TCP Ack Seq# 1
25		25	->	1.06405	2.11934	0.25449	326	272	206 40 55 19	1086	www optimal	00	HTTP [(GET /H
26		26	<-	2.11937	3.05030	0.93101	1,514	1,460	www optimal	80	206 40 55 19	1006	HTTP [(Reply H
27		27	<-	2.11949	3.26005	1.14056	1,514	1,460	www optimal	00	206 40 55 19	1006	HTTP [(Reply C
28		28	<-	2.12936	3.48536	1.35600	1,514	1,460	www optimal	80	206 40 55 19	1006	HTTP [(Reply C
29		29	<-	2.12948	3.70324	1.57376	1,514	1,460	www optimal	80	206 40 55 19	1006	HTTP [(Reply C

Ready

图 8-8 服务器与客户端之间传送包信息统计

- 线程信息统计，监控线程的内容和生存周期，以及线程与数据包的关系，如图 8-9 所示。

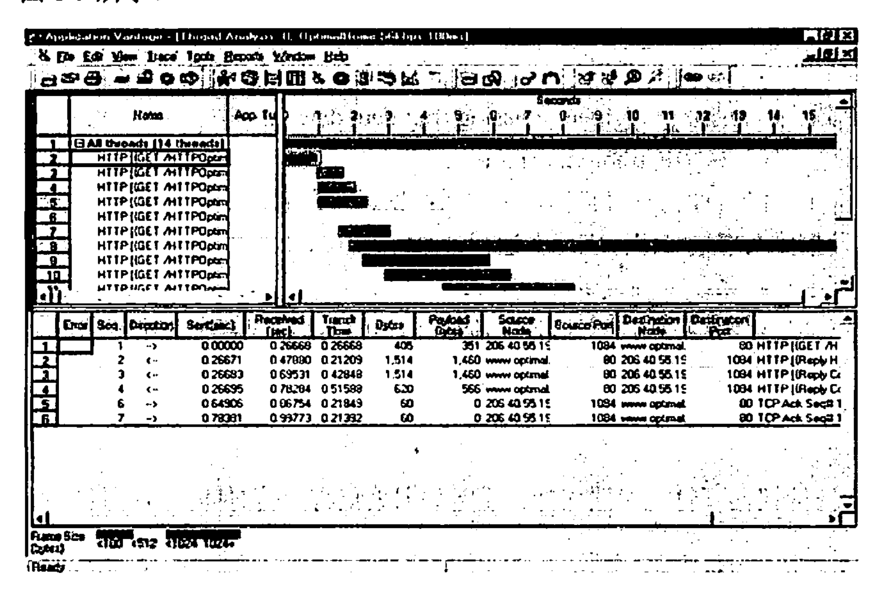


图 8-9　线程信息统计

- 负载的高峰时刻，监控到负载的平均值以及高峰值，并且高峰时刻可以与相关的线程、数据包、帧相关联，如图 8-10 所示。

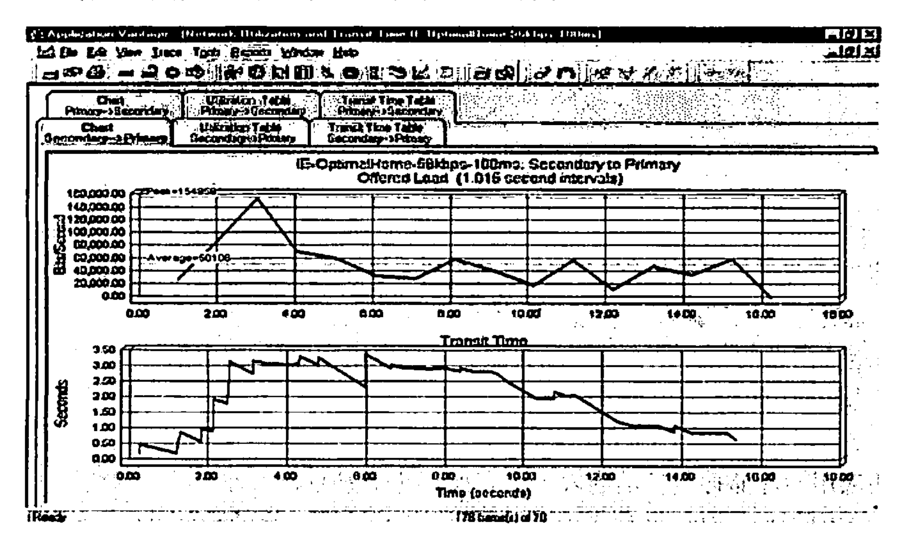


图 8-10　负载的高峰时刻

- 故障错误总结。

① 应用级错误：HTTP, FTP, DNS, FTP, No response seen, repeated DNS request, ...

② TCP 错误：retransmissions, missing data, frame out of sequence, connection errors, resets, ...

③ IP 错误：missing fragment, missing fragment data, ...

④ 其他错误：ICMP, Improper frame, ...

网络应用性能监控的测试目标，是在系统试运行之后，我们需要及时准确地了解网络上正在发生什么事情；什么应用在运行，如何运行；多少 PC 正在访问 LAN 或 WAN；哪些应用程序导致系统瓶颈或资源竞争。

利用工具进行网络应用性能监控，监控探针可以部署在整个应用范围内，如图 8-11 所示。监控工具主要包括以下部分：

① 探针。采集与存储数据，并根据应用对数据进行分类。设置的原则是根据网络组成和监控要求。

② 探针管理器。管理配置探针，设定数据采集与上传时间，合并收集的数据。

③ 时间服务器。对探针进行时针同步。

④ 交互界面。数据展示平台。

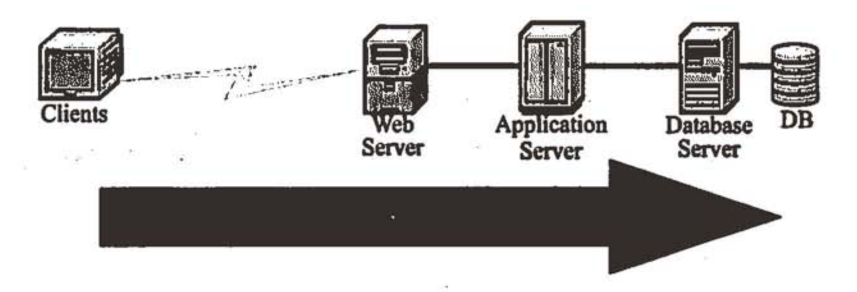


图 8-11　探针布署的应用范围

用户关心的需要进行网络监控的问题主要有以下几个方面。

- 分析关键应用程序的性能；
- 定位问题的根源是在客户端、服务器、应用程序还是网络；
- 哪些应用程序占用大量带宽；
- 哪些用户产生了最大的网络流量。

网络应用性能监控包括的内容主要有如下几个方面。

- 应用监视：1500 多种应用及 15 种定义模式、网络的硬件设备、网络应用的流量和流量的拓扑结构，如图 8-12 所示为测试工具中提供的应用监视。

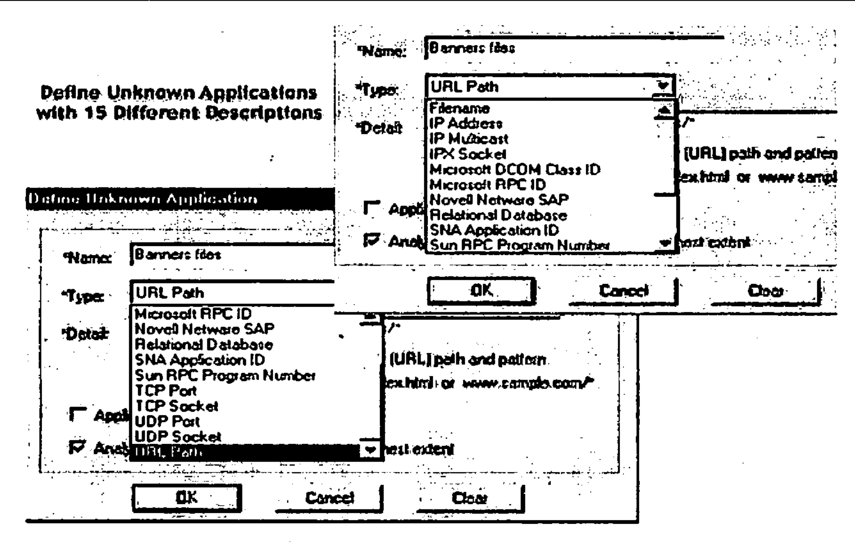


图 8-12　测试工具提供的应用监视

- 关键特性：客户和服务器通信量、应用响应时间和资源应用的业务水平等，如图 8-13 所示为工作站信息。

Station	% of Total B.	Total Bytes Sent	Multicast Bytes	Avg. Response Time	Max. Response Time	MAC Address
dwapps02.comp.com	0.54	517.66 MB	1.68 KB	6.37 ms	2.23 min	Compaq 00-00-C7
CAO-BDC	< 0.01	722.76 KB	0.00 Bytes	797.96 ms	1.05 min	
[192.168.253.2]	< 0.01	632.00 Bytes	0.00 Bytes	1.01 min	1.01 min	
Mestpdc.comp.com	0.42	403.59 MB	2.65 MB	7.61 ms	1.39 min	00-50-00-2C-9F-77
super	< 0.01	8.49 MB	0.00 Bytes	90.47 ms	1.36 min	
MALEEFH	0.01	10.90 MB	8.35 KB	5.33 sec	1.30 min	
[172.22.8.250]	< 0.01	95.84 KB	0.00 Bytes	7.06 sec	1.13 min	
Mepdc01.comp.com	0.16	157.69 MB	748.21 KB	120.10 ms	1.00 min	00-50-00-2C-66-20
[206.67.102.4]	< 0.01	5.74 MB	0.00 Bytes	197.26 ms	55.70 sec	
MA115715M	0.01	10.72 MB	7.13 KB	1.26 sec	42.66 sec	
[172.24.63.247]	< 0.01	37.81 KB	0.00 Bytes	1.31 sec	42.63 sec	
[172.22.26.251]	< 0.01	625.65 KB	0.00 Bytes	72.60 ms	35.25 sec	
fhdc1.comp.com	0.09	856.66 MB	416.78 KB	2.63 ms	34.67 sec	00-60-00-55-68-1E
dtprnt0.comp.com	1.05	1,012.03 MB	70.73 KB	6.06 ms	33.40 sec	00-50-00-44-71-93
CWBR-GRU-PR01.CO.COM	0.03	31.69 MB	0.00 Bytes	749.63 ms	31.29 sec	
MA42970SM	0.02	16.10 MB	4.49 KB	3.42 sec	31.01 sec	
FH7424AH	< 0.01	2.20 MB	2.03 KB	4.40 sec	29.56 sec	
FH2765BS	< 0.01	3.37 MB	10.29 KB	5.64 sec	25.50 sec	
CW-SINGAPORE	< 0.01	5.70 MB	0.00 Bytes	603.27 ms	24.72 sec	
CORPPAY2	0.41	393.28 MB	281.00 Bytes	2.19 ms	24.62 sec	
cwnl-ams-pr01.nl.comp.com	< 0.01	7.59 MB	0.00 Bytes	482.86 ms	20.67 sec	
MA8475KN	0.01	11.44 MB	0.12 KB	370.03 ms	20.19 sec	
FH5342ES	< 0.01	2.18 MB	17.12 KB	1.25 sec	17.96 sec	
FH2070PM	< 0.01	1.56 MB	0.00 Bytes	2.39 sec	16.45 sec	
rOhermes.programset.com	< 0.01	4.30 MB	0.00 Bytes	570.63 ms	16.37 sec	
AMELES	< 0.01	19.05 KB	0.00 Bytes	758.57 ms	15.73 sec	
MW1964	< 0.01	668.68 KB	0.00 Bytes	257.14 ms	15.16 sec	

图 8-13　工作站信息

- 按会话统计传输负载：测试应用和会话级响应时间，以及自动为通过网络中每一个联网设备的每一个应用程序生成负载图。
- 应用、会话级、事务响应时间，如图 8-14 所示为应用的响应时间等相关信息。



Application	% of Total Bytes	Bytes	Avg Response Time	Max Response Time	WorkStation
PDQ Mfg	25.16	23.69 GB	236.22 ms	20.39 min	1,47
WWW (Web)	14.86	13.99 GB	372.94 ms	7.33 min	1,49
SQL Server	13.62	12.83 GB	3.40 ms	3.96 sec	[illegible]
SMB	9.97	9.39 GB	7.59 ms	2.23 min	1,45
MSExchange-InfoStore	9.67	9.11 GB	41.24 ms	3.92 min	96
TCP	5.00	4.71 GB	-	-	82
FTP	2.35	2.21 GB	128.54 ms	3.90 min	9
MSExchange-DirStore	2.29	2.16 GB	46.97 ms	1.45 min	93
Print Server	1.94	1.83 GB	52.88 ms	1.13 min	3
Telnet	1.87	1.76 GB	238.43 ms	1.33 min	37
OracleFinancials	1.63	1.54 GB	14.24 ms	3.77 min	1
Desktop Publishing	1.55	1.46 GB	-	-	1
SMTP	1.47	1.39 GB	105.22 ms	1.15 min	1
ISO-DE	1.24	1.17 GB	-	-	-
Server MF Backup	0.93	833.64 MB	134.93 ms	4.45 sec	-
SNMP	0.68	655.81 MB	64.94 ms	11.90 sec	2,42
WWW (Secure Web)	0.64	615.19 MB	597.48 ms	3.19 min	33
MS Print Spooler	0.62	593.04 MB	16.71 ms	48.57 sec	12
ICMP	0.62	593.08 MB	-	-	2,13
Cisco Frames	0.30	293.15 MB	-	-	1
(M) UDP	0.27	256.26 MB	-	-	1
Sql files	0.21	206.65 MB	9.30 ms	57.46 sec	3
MS Networking	0.21	204.06 MB	33.14 ms	15.73 sec	91
Network News (NNTP)	0.20	195.42 MB	241.27 ms	26.41 sec	[illegible]
Pipe files	0.18	172.93 MB	12.44 ms	26.12 sec	5
pcANYWHERE	0.18	172.77 MB	204.43 ms	1.00 min	1
Domain Name Server	0.16	151.58 MB	74.18 ms	10.72 min	1,2

For Help, press F1

图 8-14 应用的响应时间

- 测试延迟是在何处被引入网络的，瓶颈在哪里。如图 8-15 所示为日应用流量相关信息，很容易定位高峰瓶颈时刻。

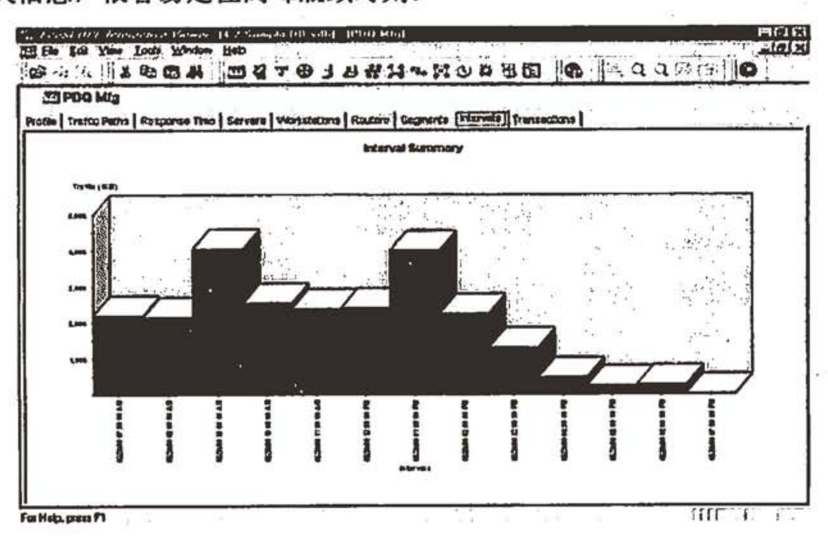


图 8-15 日应用流量相关信息

- 趋势分析。应用在网络上性能测试要得到哪些指标的值呢，以及怎样分析结果值，我们在后面章节会有详细的举例论述。

3．应用在服务器上性能的测试

这里我们谈到的"测试"的概念就是对服务器执行监控。监控的内容主要包括操作系统、数据库以及中间件等。目前监控的手段可以采用工具自动监控，也可以使用操作系统、数据库、中间件本身提供的监控工具。利用工具监控有下列优点。

- 减少故障诊断和分析时间；
- 减少手工定位的时间和避免误诊；
- 在问题发生前定位故障；
- 验证可达到的性能水平和服务水平协议；
- 持续的服务器、数据库和应用性能和可用性监控；
- 故障诊断和恢复：自动报警、故障恢复程序、故障恢复信息；
- 服务器、应用可用性和性能报告。

操作系统、数据库、中间件本身提供的监控工具有时采用命令行的方式，有时具备友好的图形界面，例如，Saloris 监控服务器资源占用可以使用 vmstat 或者 iostat 命令，Web 应用中间件 Websphere 的监控可以采用系统本身提供的 Web 页面的监控工具，当然也有一些用于特定系统的监控工具，例如用于 AIX 操作系统的监控工具 nmon32。

操作系统的监控涉及后台重要服务器操作系统监控，如果系统采用负载均衡机制，那么还有必要验证负载均衡是否能处理大的客户端压力，并且正确实现负载均衡。操作系统有很多种类型，监控的指标也不尽相同，但对于主流的操作系统，我们最关注的指标包括三个，即 CPU、内存以及硬盘，这些指标怎样分析以及对其他关联指标的影响如何，在后面章节我们会以实例的形式详细论述。

对数据库的监控非常复杂，不同数据库监控的指标存在差异，我们将共性的指标抽取出来，如下所示。

- 监控数据库系统中关键的资源；
- 监测读写页面的使用情况；
- 监控超出共享内存缓冲区的操作数；
- 监测上一轮询期间作业等待缓冲区的时间；
- 跟踪共享内存中物理日志和逻辑日志的缓冲区的使用率；
- 监控磁盘的数据块使用情况以及被频繁读写的热点区域；
- 监控用户事务或者表空间监控事务日志；
- 监控数据库锁资源；
- 监测关键业务的数据表的表空间增长；
- 监控 SQL 执行情况。

下面举一个 Oracle 资源监控的例子，可以看到重点关注的内容包括内存利用、事件



统计、SQL 分析、会话统计。

- 内存利用：

① db block gets；

② db block changes；

③ global cache gets；

④ global cache get time。

- 事件统计：

① enqueue waits；

② shared hash latch upgrades – no wait；

③ shared hash latch upgrades – wait；

④ redo log space wait time。

- SQL 分析：

① table scan rows gotten；

② table scans(long tables)；

③ table scans(short tables)；

④ index fast full scans (full)。

- 会话统计：

① session logical reads；

② session stored procedure space；

③ CPU used by this session；

④ session connect time。

中间件服务器包括 Web 服务器，例如 Apache；Web 应用服务器，例如 Websphere 和 WebLogic；应用服务器，例如 tuxedo 等。国产中间件目前也在广泛地使用，例如 TongLink、名称等。中间件是客户端负载压力的直接承受者，中间件的资源使用得是否合理，与客户端以及与后台数据库服务器连接是否合理，都直接影响系统的性能。

中间件的监控要得到哪些指标的值呢，以及怎样分析结果值，我们在后面章节会有详细的论述。

8.2.2 疲劳强度测试

疲劳强度对系统来讲也是一种负载，它强调的是长时间的考核。

1. 日常业务疲劳强度模拟

有些客户反映“为什么我们的系统运行一段时间后服务器就要重起”，还有些客户反映“为什么我们的系统越用越慢呢”等这样一类问题，这些就是系统不能承受长时间

疲劳运行的表现。

系统日常业务可能负载并不大，但其特点是持续时间长，比如正常情况下每天持续 8 小时，而对某些系统，24 小时都需要运转。那么系统能否在正常负载情况下长时间运行呢？要验证此项性能的测试就是疲劳强度测试。

日常业务疲劳强度测试，就是模拟系统的日常业务，持续执行"一段时间"，暴露系统的性能问题，例如内存泄漏、资源争用等，分析与调整的方法与并发性能测试是非常类似的。

2．高峰业务疲劳强度模拟

一般情况下系统运行都有其高峰期，比如对一个鲜花订购系统而言，在特殊的节日，例如情人节、母亲节等，都是其高峰期；那么一个医院网上挂号系统，早晨 8:00～11:00 就是其业务的高峰期。疲劳强度测试必须要模拟这样的高峰业务。我们可以看到，这样的负载是对系统的双重考验，既包括负载，又包括长时间。

这里我们需要对"一段时间"有个合理的选择，这个时间指标要满足两个主要条件，一是这段模拟时间所处理的交易量要达到系统疲劳强度需求的业务量，例如某个系统至少要无故障运行 3 个月，处理 30 万笔交易，那么我们所执行的疲劳强度测试就必须保证这个交易量，以满足系统对长时间运行过程中所递增的数据量的要求；二是在这段测试周期中必须通过加大负载，以及尽可能长的测试周期来保证疲劳强度测试。

在后面章节的实例部分有相关举例。

8.2.3　大数据量测试

1．大数据量测试类型

大数据量测试包括独立数据量测试和综合数据量测试两种主要类型。

（1）独立数据量测试

针对某些系统存储、传输、统计、查询等业务进行单用户大数据量测试。

例如，对某些系统经常会有上传、下载的操作，操作的对象可能就是大数据量，包括图片文件、音频文件或者视频文件等。还有些系统存在大量的批处理任务，批处理任务是指一次操作将对数据库中大量数据进行互斥访问的数据库事务。这种类型的事务通常将更新同一个数据库表中的数千项乃至更多的数据。由于这类任务把所有操作放置在同一个数据库事务中，所访问的资源在其执行过程中始终被锁定，必然会对其他普通事务的访问造成影响。此外，由于这类任务本身将对数据库服务器造成巨大的负担，使得服务器负载加重，从而影响独立事务的响应时间。通常情况下，批处理任务推荐在系统具有较长空闲时完成（如晚上），这样可以保证不对独立事务造成影响。如果由于业务的要求，批处理任务必须与独立事务混合运行，则必须对其加以改造，以减轻对其他事务

的影响。

（2）综合数据量测试

我们提出“一定的数据量是并发测试与疲劳测试的基础”，在并发测试和疲劳强度测试过程中，如果不考虑数据量对系统性能的影响，无疑会带来一个缺陷。例如，模拟某个系统执行“查询”操作，在“并发用户数为 100、查询记录数为 10000 条”这样的负载下，这个系统运转正常，性能可接受；但是当负载发生变化，变为“并发用户数为 100、查询记录数为 100000 条”时，系统出现长时间无响应现象。因此在测试实施过程中，我们要采用并发测试、疲劳强度测试以及大数据量测试相结合的综合测试方案。

2. 自动生成大数据量

如何解决“大数据量测试需求，但很难在较短的时间内生成大量业务数据”？

首先，可以借助自动化测试工具，利用数据库测试数据自动生成工具，例如 TESTBytes，确定需要生成的数据类型，通过与数据库的连接来自动生成数百万行的正确的测试数据。

其次，利用自动化负载压力测试工具，模拟用户业务操作，同时并发数百个或者数千个用户生成相关数据，并且测试工程师并不需要清楚地知道数据表与表之间的关系等细节内容，这样就事半功倍了。例如要生成订单，不必考虑订单中的信息在数据库内部到底与哪些表有关系，只需要简单录制一个用户生成订单的操作，然后模拟大量虚拟用户生成订单数据就可以了。

再次，我们还可以针对某个应用，在了解整个数据库结构的基础上，自主开发数据生成工具，也可以利用数据库本身提供的辅助工具来生成数据。

3. 大数据量管理

具备大数据量测试条件之后，并非就大功告成了，如何管理这些数据决定了能否成功地实现大数据量测试。可以采用手工管理和自动化工具管理两种方式。下面给大家介绍一种数据管理工具 File-Aid/CS。

File-Aid/CS 是一套为帮助开发者、测试人员、质量保证团队更加有效地在开发、测试和支持 C/S 或 Web 应用中的测试数据管理工具。File-Aid/CS 提供数据拷贝，构造子集，数据转换，数据编辑，数据浏览，数据生成，数据比较，数据迁移等功能。File-Aid/CS 运行在 Windows NT、XP、2000、98 等平台上，支持 Oracle，Microsoft SQL Server，DB2 UDB，Sybase 和 Informix 数据库。

下面举例来看这个工具有哪些用途。例如，我们需要比较一个软件的数据库表中字段格式是否与标准格式相符，可以理解为一种标准符合性测试。借助于这个工具，可以做到下面几点。

- 比较数据和数据库结构；

- 转换关系数据库数据成 XML 数据；
- 比较 XML 数据与关系数据库数据；
- 比较 XML 文件。

利用 File-Aid/CS 中提供的数据迁移功能，还可以实现大数据量跨平台迁移，例如在平台软件测试中，可以将为 Oracle 数据库准备的数据直接迁移到 SQLServer 数据库上。

8.3　负载压力测试指标

负载压力测试主要测试哪些内容，怎样才能达到测试目的，怎样又能事半功倍呢？这些问题的答案就是我们在这里讨论的测试指标。

每个测试工具都会提供许多测试指标，有些是大家都具备的，例如交易响应时间，而有些是某个工具独具的“招牌”，例如 LoadRunner 丰富的资源监控计数器。在选择工具时可以综合考虑其优劣，关键在于能够提供满足测试目的指标的工具，这才是我们要选择的工具。一般情况下，我们可以选择的指标包括以下几类：

- 客户端交易处理性能指标；
- 服务器资源监控指标，例如：UNIX；
- 数据库资源监控指标，例如：Oracle；
- Web 服务器监控指标，例如：Apache；
- 中间件监控指标，例如：TUXEDO 等。

要提醒大家的是，指标的选择并非越多越好。如果测试需求仅仅是系统的负载压力性能评测，那么只要重点关注客户端交易处理性能指标就够了，许多工具都可以帮助你完成任务；如果测试需求关心系统后台服务器承受负载压力的能力，那么我们重点就是监控服务器、数据库、中间件等的资源使用情况；较高级别的性能调优是在系统故障定位的前提下实施的，故障定位的过程同时考验测试工程师的素质与测试工具的能力，可以说是非常困难的一件事情。这种情况下，我们要测试的指标就要求非常全面，包括客户端交易处理性能指标、服务器资源监控指标、网络品质指标，以及应用在网络上的故障定位指标等。这里包含了两层含义，一层是系统的负载压力故障定位，另一层是系统应用实际部署故障定位。前一层是后一层的基础。需要说明的是，要完成这一工作，有时需要利用多种类型的测试工具，例如：负载压力测试工具、网络品质监控工具、应用网络故障定位工具、服务器资源监控工具等。

8.3.1　交易处理性能指标

交易处理性能指标主要包括下列 4 项。



- 并发用户数指标。
- 交易处理指标。

① 平均事务响应时间。
② 每秒事务数。
③ 每秒事务总数。
④ 事务摘要。
⑤ 事务性能摘要。
⑥ 事务响应时间（负载下）。
⑦ 事务响应时间（百分比）。
⑧ 事务响应时间（分布）。

- Web 请求指标。

① 每秒点击次数。
② 点击次数摘要。
③ 吞吐量。
④ 吞吐量摘要。
⑤ HTTP 状态代码摘要。
⑥ 每秒 HTTP 响应数。
⑦ 每秒下载页面数。
⑧ 每秒重试次数。
⑨ 重试次数摘要。
⑩ 连接数。
⑪ 每秒连接数。
⑫ 每秒 SSL 连接数。

- Web 页面组件指标。

① 激活网页细分。
② 页面组件细分。
③ 页面组件细分（随时间变化）。
④ 页面下载时间细分。
⑤ 页面下载时间细分（随时间变化）。
⑥ 第一次缓冲细分时间。
⑦ 第一次缓冲时间细分（随时间变化）。
⑧ 已下载组件大小。

8.3.2 服务器操作系统资源监控

1. UNIX 操作系统

如表 8-1 所示为 UNIX 操作系统资源监控指标。

表 8-1 UNIX 操作系统资源监控指标

度 量	描 述
Average load	上一分钟同时处于“就位”状态的平均进程数
Collision rate	每秒钟在以太网上检测到的冲突数
Context switches rate	每秒钟在进程或线程之间的切换次数
CPU utilization	CPU 的使用时间百分比
Disk rate	磁盘传输速率
Incoming packets error rate	接收以太网数据包时每秒钟接收到的错误数
Incoming packets rate	每秒钟传入的以太网数据包数
Interrupt rate	每秒钟内的设备中断数
Outgoing packets errors rate	发送以太网数据包时每秒钟发送的错误数
Outgoing packets rate	每秒钟传出的以太网数据包数
Page-in rate	每秒钟读入到物理内存中的页数
Page-out rate	每秒钟写入页面文件和从物理内存中删除的页数
Paging rate	每秒钟读入物理内存或写入页面文件中的页数
Swap-in rate	正在交换的进程数
Swap-out rate	正在交换的进程数
System mode CPU utilization	在系统模式下使用 CPU 的时间百分比
User mode CPU utilization	在用户模式下使用 CPU 的时间百分比

Linux 操作系统与 UNIX 操作系统类似。

2. Windows 操作系统

如表 8-2 所示为 Windows 操作系统资源监控指标。

表 8-2 Windows 操作系统资源监控指标

对 象	度 量	描 述
System	% Total Processor Time	系统上所有处理器都忙于执行非空闲线程的平均时间的百分比。在多处理器系统上，如果所有处理器始终繁忙，此值是 100%，如果所有处理器是 50%繁忙，此值为 50%。而如果这些处理器中的 1/4 是 100%繁忙的，则此值为 25%。它反映了用于有用作业上的时间的比率。每个处理器将分配给闲置进程中的一个闲置线程，以消耗所有其他线程不使用的那些非生产性处理器周期



续表

对　　象	度　　量	描　　述
Processor	% Processor Time(Windows 2000)	处理器执行非空闲线程的时间百分比，此计数器设计为处理器活动的一个主要指示器。它是通过测量处理器在每个采样间隔中执行空闲进程的线程所花费的时间，然后从100%中减去此值来进行计算的（每个处理器都有一个空闲线程，它在没有其他线程准备运行时消耗处理器周期）。它可以反映有用作业占用的采样间隔的百分比。此计数器显示在采样期间所观察到的繁忙时间的平均百分比。它是通过监视服务处于非活动状态的时间，然后从100%中减去此值来计算的
System	File Data Operation/sec	计算机向文件系统设备发出读取和写入操作的速度。此操作不包括文件控制操作
System	Processor Queue Length	线程单元中的处理器队列的即时长度。如果您不同时监视线程计数，则此计数始终为0。所有处理器都使用单一队列（线程在该队列中等待处理器进行循环）。此长度不包括当前正在执行的线程。一般情况下，如果处理器队列的长度一直超过2，则可能表示处理器堵塞。此值为即时计数，不是一段时间的平均值
Memory	Page Faults/sec	此值为处理器中的页面错误的计数。当进程引用特定的虚拟内存页，该页不在其在主内存的工作集当中时，将出现页面错误。如果某页位于待机列表中（因此它已经位于主内存中），或者它正在被共享该页的其他进程所使用，则页面错误不会导致该页从磁盘中提取出
PhysicalDisk	% Disk Time	选定的磁盘驱动器对读写请求提供服务的已用时间所占百分比
Memory	Pool Nonpaged Bytes	非分页池中的字节数，指可供操作系统组件完成指定任务后从其中获得空间的系统内存区域。非分页池页面不可以退出到分页文件中，它们自分配以来就始终位于主内存中
Memory	Pages/sec	为解析内存对页面（引用时不在内存中）的引用而从磁盘读取的页数或写入磁盘的页数。这是“Pages Input/sec”和“Pages Output/sec”的和。此计数器中包括的页面流量代表着用于访问应用程序的文件数据的系统缓存。此值还包括传递到/来自非缓存映射内存文件的页数，如果您关心内存压力过大问题（即系统失效）和可能产生的过多分页，则这是您值得考虑的主要计数器

续表

对　象	度　量	描　述
System	Total Interrupts/sec	计算机接收并处理硬件中断的速度。可能生成中断的设备有系统时钟、鼠标、数据通信线路、网络接口卡和其他外围设备。此计数指示这些设备在计算机上所处的繁忙程度。另请参阅 Processor: Interrupts/sec
Objects	Threads	计算机在收集数据时的线程数。注意，这是一个即时计数，不是一段时间的平均值。线程是基本的可执行实体，用于在处理器中执行指令
Process	Private Bytes	专为此进程分配，无法与其他进程共享的当前字节数

针对操作系统的监控，如果我们需要监控磁盘管理、文件系统、内存、CPU 等方面的内容，下面给出相关的一些监控建议。

- 磁盘管理。

① 采集物理读/写和逻辑读/写的信息。

② 收集操作系统和其他平台上的磁盘忙信息。

③ 监控 I/O。

- 文件系统。

① 显示每个文件系统的使用率，检测文件系统空闲空间的大小。

② 剪裁文件系统——删除指定的 CORE 文件和其他文件。

③ 显示文件系统的 mount on device、type、size 等内容。

④ 可以监控特殊的文件系统，如 NFS，CD-ROM。

⑤ 检测特定文件的存在及超出特定期限的文件存在。

- 内存。

① 显示可用的内存数量。

② 决定当前的内存短缺量。

③ 帮助分析内存问题。

④ 显示内存的实存、所有虚存和 kernel 的状态等信息。

- CPU。

① 记录 CPU 的使用率。

② 监测 CPU 参数，包括 CPU idle，CPU waits，CPU system usage，CPU user usage，run queue length。

③ 显示 CPU context switches 的总数。

④ 显示 CPU 处理系统任务和完成用户任务的时间比例。



8.3.3 数据库资源监控

1. Oracle

如表 8-3 所示为 Oracle 资源监控指标。

表 8-3 Oracle 资源监控指标

度 量	描 述
CPU used by this session	这是在用户调用开始和结束之间会话所占用的 CPU 时间（以 10ms 为单位）。一些用户调用在 10ms 之内即可完成，因此用户调用的开始和结束时间可以是相同的，在这种情况下，系统值为 0ms，操作系统报告中可能有类似的问题，尤其是在经历许多上下文切换的系统中
Bytes received via SQL*Net from client	通过 Net8 从客户端接收的总字节数
Logons current	当前的登录总数
Opens of replaced files	由于已经不在进程文件缓存中，所以需要重新打开的文件总数
User calls	在每次登录、解析或执行时，Oracle 会分配资源（call state 对象）以记录相关的用户调用数据结构。在确定活动时，用户调用与 RPI 调用的比指明了因用户发往 Oracle 的请求类型而生成的内部工作量
SQL*Net roundtrips to/from client	发送到客户端和从客户端接受的 Net8 消息的总数
Bytes sent via SQL*Net to client	从前台进程中发送到客户端的总字节数
Opened cursors current	当前打开的光标总数
DB block changes	由于与一致更改的关系非常密切，此统计计算对 SGA 中所有块执行的、作为更新或删除操作一部分的更改总数。这些更改将生成重做日志项。如果事物被提交，将是对数据库的永久性更改。此统计是一个全部数据库作业的粗略指示，并且指出（可能在每事物级上）弄脏缓冲区的速率
Total file opens	由实例执行的文件打开总数，每个进程需要许多文件（控制文件、日志文件、数据库文件）以便针对数据库进行工作

2. Sysbase

如表 8-4 所示为 Sysbase 资源监控指标。

表 8-4 Sysbase 资源监控指标

对 象	度 量	描 述
Network	Average packet size (Read)	报告接收的网络数据包数
	Average packet size (Send)	报告发送的网络数据包数
	Network bytes (Read)	报告在采样间隔期间接收的字节数

续表

对　　象	度　　量	描　　述
Network	Network bytes (Read)/sec	报告每秒接收的字节数
	Network bytes (Send)	报告在采样间隔期间发送的字节数
	Network bytes (Send)/sec	报告每秒发送的字节数
	Network packets (Read)	报告在采样间隔期间接收的网络数据包数
	Network packets (Read)/sec	报告每秒接收的网络数据包数
	Network packets (Send)	报告在采样间隔期间发送的网络数据包数
	Network packets (Send)/sec	报告每秒发送的网络数据包数
Memory	Memory	报告分配给页面缓存的内存量（以字节为单位）
Disk	Reads	报告从数据库设备中进行的读取数
	Writes	报告从数据库设备中进行的写入数
	Waits	报告访问设备的等待次数
	Grants	报告授予访问设备权限的次数
Engine	Server is busy (%)	报告 Adaptive Server 处于“繁忙”状态的时间百分比
	CPU time	报告引擎使用了多少“繁忙”时间
	Logical pages (Read)	报告从缓存和从数据库服务中的数据页读取数
	Pages from disk (Read)	报告从数据缓存中无法获得的数据页读取数
	Pages stored	报告写入数据库设备的数据页数
Stored Procedures	Executed (sampling period)	报告在采样间隔期间执行存储过程的次数
	Executed (session)	报告在会话期间执行存储过程的次数
	Average duration (sampling period)	报告在采样间隔期间执行存储过程所花费的时间（以秒为单位）
	Average duration (session)	报告在会话期间执行存储过程所花费的时间（以秒为单位）
Locks	% Requests	报告成功锁定请求的百分比
	Locks count	报告锁定数。这是一个累加值
	Granted immediately	报告立即授予（而不必等待释放另一个锁定）的锁定数
	Granted after wait	报告在另一个锁定被释放后授予的锁定数
	Not granted	报告已经请求但是没有授予的锁定数
	Wait time (avg.)	报告等待锁定的平均时间
SqlSrvr	Locks/sec	报告锁定数，这是一个累加值
	%Processor time (server)	报告 Adaptive server 处于“繁忙”状态的时间百分比
	Transactions	报告已提交的 Transact-SQL 语句块（事务）数
	Deadlocks	报告死锁数



续表

对　象	度　量	描　述
Cache	%Hits	报告从缓存中得到的数据页读取（而无需从物理页读取）次数百分比
	Pages (Read)	报告从缓存和从数据库服务中的数据页读取数
Cache	Pages (Read)/sec	报告每秒从缓存和从数据库服务中的数据页读取数
	Pages from disk (Read)	报告从数据缓存中无法获得的数据页读取数
	Pages from disk (Read) /sec	报告每秒从数据缓存中无法获得的数据页读取数
	Pages (Write)	报告写入数据库设备的数据页数
	Pages (Write)/sec	报告每秒写入数据库设备的数据页数
Process	% Process time (process)	报告运行特定应用程序的进程处于“运行”状态的时间百分比（多于所有进程都处于“运行”状态的时间）
	Locks/sec	报告各进程的锁定数，这是一个累加值
	%Cache hit	报告各进程从缓存中得到的数据页读取（而无需从物理页读取）次数百分比
	Pages (Write)	报告各进程写入数据库设备的数据页数
Transaction	Transaction	报告在会话期间已提交的 Transact－SQL 语句块（事务）数
Transaction	Rows(Deleted)	报告在会话期间从数据库表中删除的行数
	Inserts	报告在会话期间到数据库表中的插入操作数
	Updates	报告在会话期间对数据库表所做的更新
	Updates in place	报告在会话期间昂贵的就地和非就地更新（除了已推迟的更新之外的所有更新）的总和
	Transactions/sec	报告每秒提交的 Transact-SQL 语句块（事务）数
	Rows (Deleted)/sec	报告每秒从数据库表中删除的行数
	Inserts/sec	报告每秒到数据库表中的插入操作数
	Updates/sec	报告每秒对数据库表所做的更新
	Updates in place/sec	报告每秒代价高昂的就地和非就地更新（除了已推迟的更新之外的所有更新）的总和

3．DB2

如表 8-5、表 8-6 及表 8-7 所示为 DB2 资源监控指标。

表 8-5　DB2 资源监控指标（数据库管理）

度　量	描　述
rem_cons_in	到正在被监视的数据库管理器实例的当前连接数，从远程客户端启动
rem_cons_in_exec	当前连接到数据库的远程应用程序数，这些应用程序正在处理被监视数据库管理器实例内的工作单元
local_cons	当前连接到被监视数据库管理器实例内的数据库的本地应用程序数

续表

度　量	描　述
local_cons_in_exec	当前连接到被监视数据库管理器实例内的数据库并正在处理工作单元的本地应用程序数
con_local_dbases	与应用程序相连接的本地数据库数
agents_registered	在被监视数据库管理器实例中注册的代理程序（协调程序代理程序和子代理程序）数
agents_waiting_on_token	等待令牌以在数据库管理器中执行事物的代理程序数
idle_agents	代理程序池中当前未分配给应用程序，因此仍处于“空闲”状态的代理程序数
agents_from_pool	代理程序池中已分配的代理程序数
agents_created_empty_pool	由于代理程序池是空的而创建的代理程序数
agents_stolen	从应用程序中盗用代理程序的次数。重新分配与应用程序相关联的空闲代理程序，以便对其他应用程序执行操作，称作“盗用”
comm._private_mem	在快照时，数据库管理器实例当前已经提交的专用内存量
inactive_gw_agents	DRDA 连接池中，准备好与 DRDA 数据库的连接，但尚未活动的 DRDA 代理程序数
num_gw_conn_switches	代理程序池中，代理程序已准备好连接但又被其他 DRDA 数据库盗用的次数
sort_heap_allocated	拍快照时，以所选择的级别为所有排序分配的排序堆空间的总页数
post_threshold_sorts	达到排序堆阈值后，已请求的堆的排序数
piped_sorts_requested	已经请求的管道排序数
piped_sorts_accepted	已经接受的管道排序数

表 8-6　DB2 资源监控指标（数据库）

度　量	描　述
appls_cur_cons	当前已连接到数据库的应用程序数
appls_in_db2	当前已连接到数据库并且数据库管理器当前正在处理其请求的应用程序数
total_sec_cons	由子代理程序创建的到节点上数据库的连接数
num_assoc_agents	在应用程序级，这是与应用程序关联的子代理程序数；在数据库级，它是所有应用程序的子代理程序数
sort_heap_allocated	拍快照时，以所选择的级别为所有排序分配的排序堆空间的总页数
total_sorts	已经执行的排序总数
total_sort_time	所有已执行排序的总已用时间（以毫秒为单位）
sort_overflows	用完排序堆并且可能需要临时磁盘存储空间的排序总数
active_sorts	数据库中当前已经分配了排序堆的排序数
total_hash_joins	执行的哈希连接的总数



续表

度　量	描　述
total_hash_loops	哈希连接的单一分区大于可用的排序堆空间的总次数
hash_join_overflows	哈希连接数据大小超过可用排序堆空间的次数
hash_join_small_overflows	哈希连接数据大小超过可用排序堆空间，但超出比率小于10%的次数
pool_data_l_reads	已经通过缓冲池的数据页逻辑读取请求数
pool_data_p_reads	要求I/O将数据页放入缓冲池的读取请求数
pool_data_writes	将缓冲池数据页物理地写入磁盘的次数
pool_index_l_reads	已经通过缓冲池的索引页逻辑读取请求数
pool_index_p_reads	需要将索引页放入缓冲池的物理读取请求数
pool_index_writes	将缓冲池中的索引页物理地写入磁盘的次数
pool_read_time	处理读取请求（使数据或索引页从磁盘物理地读入缓冲池）的总已用时间
pool_write_time	从缓冲池中将数据或索引页物理地写入磁盘所花费的总时间
files_closed	已关闭的数据库文件的总数
pool_async_data_reads	异步读入到缓冲池中的页数
pool_async_data_writes	使用异步页清理器或预取器，将缓冲池索引页物理地写入磁盘的次数。预取器可能已经将脏页写入磁盘，从而为预取页腾出空间
pool_async_index_writes	使用异步页清理器或预取器，将缓冲池索引页物理地写入磁盘的次数。预取器可能已经将脏页写入磁盘，从而为预取页腾出空间
pool_async_index_reads	由预取器异步读入到缓冲池中的索引页数
pool_async_read_time	数据库管理器预取器花在读取操作上的总已用时间
pool_async_write_time	数据库管理器页清理器从缓存池中将数据或索引页写入磁盘的总已用时间
pool_async_data_read_reqs	异步读取请求数
pool_lsn_gap_clns	由于所用的记录空间已经到达数据库的预定义标准而调用页清理器的次数
pool_drty_pg_steal_clns	由于在受损缓冲区代替数据库期间需要进行同步写入而调用页清理器的次数
pool_drty_pg_thrsh_clns	由于缓冲池已经到达数据库的脏页阈值标准而调用页清理器的次数
prefetch_wait_time	应用程序等待I/O服务器（预取器）将页加载到缓冲池所花费的时间
pool_date_to_estore	复制到扩展存储区的缓冲池数据页数
pool_index_to_estore	复制到扩展存储区的缓冲池索引页数
pool_data_from_estore	从扩展存储区复制的缓冲池数据页数
pool_index_from_estore	从扩展存储区复制的缓冲池索引页数
direct_reads	不使用缓冲池的读取操作数
direct_writes	不使用缓冲池的写入操作数
x_lock_escals	从几行锁定上升为一个排他表格锁定的次数，或者一行的排他锁定使表格锁定变为排他锁定的次数

续表

度　　量	描　　述
lock_timeouts	锁定对象的请求因超时而未能得到满足的次数
lock_waits	应用程序或连接等待锁定的总次数
lock_wait_time	等待锁定的总已用时间
locks_waiting	等待锁定的代理程序数
rows_deleted	试图删除行的次数
rows_inserted	试图插入行的次数
rows_updated	试图更新行的次数
rows_selected	被选择并返回到应用程序的行数
int_rows_deleted	作为内部活动的结果从数据库删除的行数
int_rows_updated	作为内部活动的结果从数据库更新的行数
int_rows_inserted	作为由触发器引发的内部活动的结果插入到数据库的行数
static_sql_stmts	试图执行的静态 SQL 语句数
dynamic_sql_stmts	试图执行的动态 SQL 语句数
failed_sql_stmts	试图执行而失败的 SQL 语句数
commit_sql_stmts	试图执行的 SQL COMMIT 语句的总数
rollback_sql_stmts	试图执行的 SQL ROLLBACK 语句的总数
select_sql_stmts	已经执行的 SQL SELECT 语句数
uid_sql_stmts	已经执行的 SQL UPDATE、INSERT 和 DELETE 语句数
ddl_sql_stmts	已经执行的 SQL 数据定义语言（DDL）语句数
int_auto_rebinds	试图执行的自动重新绑定（或重新编译）数
int_commits	由数据库管理器在内部启动的提交总数
int_rollbacks	由数据库管理器在内部启动的回滚总数
int_deadlock_rollbacks	由于死锁而由数据库管理器启动的强制回滚总数。在由数据库管理器所选择应用程序的当前工作单元上执行回滚以解开死锁
binds_precompiles	试图执行的绑定和预编译数

表 8-7　DB2 资源监控指标（应用程序）

度　　量	描　　述
agents_stolen	从应用程序盗用代理程序的次数，重新分配与应用程序相关联的空闲代理程序，以便对其他应用程序执行操作，称作“盗用”
num_assoc_agents	在应用程序级，这是与应用程序关联的子代理程序数；在数据库级，它是所有应用程序的子代理程序数
total_sorts	已经执行的排序总数
total_sort_time	已经执行所有排序的总已用时间（以毫秒为单位）
sort_overflows	用完排序堆并且可能需要临时磁盘存储空间的排序总数



续表

度　量	描　述
total_hash_joins	执行的哈希连接的总数
total_hash_loops	哈希连接的单一分区大于可用的排序堆空间的总次数
hash_join_overflows	哈希连接数据大小超过可用排序堆空间的次数
hash_join_small_overflows	哈希连接数据大小超过可用排序堆空间，但超出比率小于10%的次数
pool_data_l_reads	已经通过缓冲池的数据页逻辑读取请求数
pool_data_p_reads	要求I/O将数据页放入缓冲池的读取请求数
pool_data_writes	将缓冲池数据页物理地写入磁盘的次数
pool_index_l_reads	已经通过缓冲池的索引页逻辑读取请求数
pool_index_p_reads	需要将索引页放入缓冲池的物理读取请求数
pool_index_writes	将缓冲池中的索引页物理地写入磁盘的次数
uow_log_space_used	被监视应用程序的当前工作单元使用的日志空间量（以字节为单位）
locks_held	当前保持的锁定数
deadlocks	已经发生的死锁的总数
lock_escals	从几行锁定上升为表格锁定的次数
x_lock_escals	从几行锁定上升为一个排他表格锁定的次数或者一行的排他锁定使表格锁定变为排他锁定的次数
lock_timeouts	锁定对象的请求因超时而未得到满足的次数
lock_waits	应用程序或连接等待锁定的总次数
lock_wait_time	等待锁定的总已用时间
locks_waiting	等待锁定的代理程序数
uow_lock_wait_time	此工作单元等待锁定的总已用时间
rows_deleted	试图删除行的次数
rows_inserted	试图插入行的次数
rows_updated	试图更新行的次数
rows_selected	被选择并返回到应用程序的行数
rows_written	表格中已经更改（插入、删除或更新）的行数
rows_read	从表格中读取的行数
int_rows_deleted	作为内部活动的结果从数据库删除的行数
int_rows_updated	作为内部活动的结果从数据库更新的行数
int_rows_inserted	作为由触发器引发的内部活动的结果插入到数据库的行数
open_rem_curs_	当前为此应用程序打开的远程光标数，包括由"open_rem_curs_blk"统计的那些光标
open_rem_curs_blk	当前为此应用程序打开的远程块状光标数
rej_curs_blk	拒绝服务器上I/O块的请求和将请求转换成非块的I/O请求的次数
acc_curs_blk	接受I/O块请求的次数

续表

度　量	描　述
open_loc_curs	当前为此应用程序打开的本地光标数，包括由“open_loc_curs_blk”统计的那些光标
open_loc_curs_blk	当前为此应用程序打开的本地块状光标数
static_sql_stmts	试图执行的静态 SQL 语句数
dynamic_sql_stmts	试图执行的动态 SQL 语句数
failed_sql_stmts	试图执行而失败的 SQL 语句数
commit_sql_stmts	试图执行的 SQL COMMIT 语句的总数
rollback_sql_stmts	试图执行的 SQL ROLLBACK 语句的总数
select_sql_stmts	已经执行的 SQL SELECT 语句数
uid_sql_stmts	已经执行的 SQL UPDATE、INSERT 和 DELETE 语句数
ddl_sql_stmts	已经执行的 SQL 数据定义语言（DDL）语句数
int_auto_rebinds	试图执行的自动重新绑定（或重新编译）数
int_commits	由数据库管理器在内部启动的提交总数
int_rollbacks	由数据库管理器在内部启动的回滚总数
int_deadlock_rollbacks	由于死锁而由数据库管理器启动的强制回滚总数，在由数据库管理器所选择应用程序的当前工作单元上执行回滚以解开死锁
binds_precompiles	试图执行的绑定和预编译数

4．SQL Server

如表 8-8 所示为 SQL Server 资源监控指标。

表 8-8　SQL Server 资源监控指标

度　量	描　述
% Total Processor Time (NT)	系统上所有处理器都忙于执行非空闲线程的时间的平均百分比。在多处理器系统上，如果所有处理器始终繁忙，此值是 100%；如果所有处理器是 50%繁忙，此值为 50%；而如果这些处理器中的 1/4 是 100%繁忙的，则此值为 25%，它反映了用于有用作业上的时间的比率。每个处理器将分配给空闲进程中的一个空闲线程，以消耗所有其他线程都不使用的那些非生产性处理器周期
Cache Hit Ratio	在数据缓存中找到（而不是从磁盘读取）所请求数据页的时间百分比
I/O-Batch Writes/sec	使用 Batch I/O，每秒写入磁盘的页数（以 2k 页为单位）Batch I/O 主要用于检查点线程
I/O-Lazy Writes/sec	每秒由 Lazy Writer 刷新到磁盘的页数（以 2k 页为单位）
I/O-Outstanding reads	挂起的物理读取数
I/O-Outstanding writes	挂起的物理写入数
I/O-Page Reads/sec	每秒物理页读取数



续表

度　　量	描　　述
I/O-Transactions/sec	每秒执行的 Transact-SQL 命令批处理数
User Connections	打开的用户连续数
& Processor Time (Win 2000)	处理器执行非空闲线程的时间百分比。此计数器设计为处理器活动的一个主要指示器。它是通过测量处理器在每个采样间隔中执行空闲进程的线程所花费的时间，然后从 100%中减去此时间值来进行计算的(每个处理器都有一个空闲线程，它在没有其他线程准备运行时消耗处理器周期)。它可以反映有用作业占用的采样间隔的百分比，此计数器显示在示例采样期间所观察的繁忙时间的平均百分比。它是通过监视服务处于非活动状态的时间，然后从 100%中减去此值来计算的

针对数据库系统的监控，如果我们需要监控共享内存缓冲区、会话、磁盘等方面的内容，下面给出了相关的一些监控建议。

- 监控超出共享内存缓冲区的操作数。

监控超出共享内存缓冲区的操作数，按照这个基准，我们可以对缓冲区进行额外的调整，以便更好地支持实际系统的运行需要，提高用户生产效率。

- 扩展的会话/用户检查以及参数控制。

利用扩展的会话/用户检查以及参数控制功能，向我们发出警报，帮助我们发现过多的不合理顺序扫描操作。根据这些信息，我们可以分配附加的资源，或者要求修改其应用程序，以降低对系统资源的要求。我们可以制定更精确的资源容量计划，以便实现现在和将来的业务运行需要。

- 磁盘。

监控磁盘的数据块使用情况以及被频繁读写的热点区域。通过这些信息我们可以较容易地平衡在磁盘上数据量的存储分配以及磁盘的 I/O 分配，有效地帮助我们作好磁盘容量规划，并在当前的磁盘设备上有效地提高数据的读写效率。

8.3.4 Web 服务器监控

1. Apache

如表 8-9 所示为 Apache 资源监控指标。

表 8-9 Apache 资源监控指标

度　　量	描　　述
# Busy Servers	处于繁忙状态的服务器数
# Idle Servers	处于空闲状态的服务器数
Apache CPU usage	Apache 服务器利用 CPU 的时间百分比

续表

度　量	描　述
Hits/sec	HTTP 请求速率
Kbytes Sent/sec	从 Web 服务器发送数据字节的速率

2．IIS

如表 8-10 所示为 IIS 资源监控指标。

表 8-10　IIS 资源监控指标

对　象	度　量	描　述
Web Service	Bytes Sent/sec	Web Service 发送数据字节的速率
Web Service	Bytes Received/sec	Web Service 接收数据字节的速率
Web Service	Get Requests/sec	使用 GET 方法进行 HTTP 请求的速率。尽管 GET 请求可以用于窗体，但通常用于基本文件检索或图像映射
Web Service	Post Requests/sec	使用 POST 方法进行 HTTP 请求的速率。POST 请求通常用于窗体或网关请求
Web Service	Maximum Connections	同时与 Web Service 建立的最大连接数
Web Service	Current Connections	当前与 Web Service 建立的连接数
Web Service	Current NonAnonymous Users	当前使用 Web Service 非匿名连接的用户数
Web Service	Not Found Errors/sec	由于找不到请求的文档，服务器不能满足请求而出现的错误率，这些通常作为 HTTP 404 错误代码报告到客户端
Process	Private Bytes	已经由进程分配但无法与其他进程共享的当前字节数

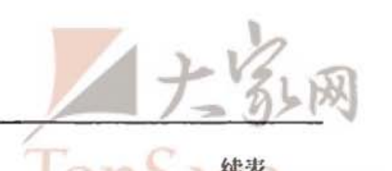
8.3.5　中间件服务器监控

1．TUXEDO

如表 8-11 所示为 TUXEDO 资源监控指标。

表 8-11　TUXEDO 资源监控指标

监 视 器	度量及描述
服务器	Requests per second：每秒钟处理的服务器请求数
	Workload per second：该工作负荷是服务器请求的加权度量。某些请求可能与其他请求有不同的权重。在默认情况下，工作负荷总是请求数的 50 倍
计算机	Workload completed per second：计算机所有服务器每单位时间完成的总工作负荷
	Workload initiated per second：计算机所有服务器每单位时间开始的总工作负荷
	Current Accessers：当前直接在该计算机上访问应用程序或通过该计算机上的工作站处理程序访问应用程序的客户端和服务器数



续表

监 视 器	度量及描述
计算机	Current Clients：当前登录到该计算机的客户端数，包括本地计算机和工作站
	Current Transactions：该计算机上正在使用的事务表项目数
队列	Bytes on queue：正在队列中等待的所有消息的总字节数
	Messages on queue：队列中正在等待的总请求数。默认情况为 0
工作站处理程序（WSH）	Bytes received per second：工作站处理程序每单位时间接收到的总字节数
	Bytes sent per second：工作站处理程序每单位时间发送回客户端的总字节数
	Messages received per second：工作站处理程序每单位时间接收到的消息数
	Messages sent per second：工作站处理程序每单位时间发送回客户端的消息数
	Number of queue blocks per second：工作站处理程序每单位时间阻止队列的次数。通过它可以了解工作站处理程序过载的频率

2．WebSphere

如表 8-12 与表 8-13 所示为 WebSphere 资源监控指标。

表 8-12　WebSphere 资源监控指标（队列性能计数器）

度 量	描 述
Event-Queue Depth High (events per second)	队列深度达到配置的最大深度时触发的事件
Event-Queue Depth Low (events per second)	队列深度达到配置的最小深度时触发的事件
Event-Queue Full (events per second)	试图将消息放到已满的队列时触发的事件
Event-Queue Service Interval High (events per second)	在超时阈值内没有消息放到队列或者没有从队列检索到消息时触发的事件
Event-Queue Service Interval OK (events per second)	在超时阈值内消息已经放到队列或者已经从队列检索到消息时触发的事件
Status-Current Depth	本地队列上的当前消息计数，该度量只适用于监视队列管理器的本地队列
Status-Open Input Count	打开的输入句柄的当前计数。将打开输入句柄，以便应用程序可以将消息"放到"队列
Status-Open Output Count	打开的输出句柄的当前计数，将打开输出句柄，以便应用程序可以从队列中"获得"消息

表 8-13　WebSphere 资源监控指标（通道性能计数器）

度 量	描 述
Event-Channel Activated (events per second)	当正等待激活，但却由于缺少队列管理器通道插槽而不能激活的通道，在由于突然可以使用通道插槽而激活时生成的事件
Event-Channel Not Activated (events per second)	当通道试图激活，但却由于缺少队列管理器通道插槽而不能激活时生成的事件

续表

度　量	描　述
Event-Channel Started (events per second)	启动通道时生成的事件
Event-Channel Stopped (events per second)	停止通道（无论停止源如何）时生成的事件
Event-Channel Stopped by User (events per second)	由用户停止通道时生成的事件
Status-Channel State	通道的当前状态。通道从停止（非活动状态）到运行（完全活动状态）经过数个状态。通道状态范围从 0（停止）～6（运行）
Status-Messages Transferred	已在通道上发送的消息的计数。如果在通道上没有发生通信，则该度量将是 0；如果队列管理器启动后没有启动该通道，则度量不可用
Status-Buffer Received	已在通道上接收的缓冲区的计数，如果在通道上没有发生通信，则该度量将是 0；如果队列管理器启动后没有启动该通道，则度量不可用
Status-Buffer Sent	已在通道上发送的缓冲区的计数，如果在通道上没有发生通信，则该度量将是 0；如果队列管理器启动后没有启动该通道，则度量不可用
Status-Bytes Received	已在通道上接收的字节的计数，如果在通道上没有发生通信，则该度量将是 0；如果队列管理器启动后没有启动该通道，则度量不可用
Status-Bytes Sent	已在通道上发送的字节的计数。如果在通道上没有发生通信，则该度量将是 0；如果队列管理器启动后没有启动该通道，则度量不可用

3. WebLogic

如表 8-14 与表 8-15 所示为 WebLogic 资源监控指标。

表 8-14　WebLogic 资源监控指标（LogBaroadcasterRuntime）

度　量	描　述
MessagesLogged	该 WebLogic 服务器实例生成的日志消息总数
Registered	如果已取消注册该对象表示的 Mbean，则返回“False”
CachingDisabled	禁用代理中的缓存的专用属性

表 8-15　WebLogic 资源监控指标（ServerSecurityRuntime）

度　量	描　述
UnlockedUsersTotalCount	返回在服务器上取消锁定用户的次数
InvalidLoginUsersHighCount	返回具有显著的无效服务器登录尝试的用户的最大数目
LoginAttemptsWhilelockedTotalCount	返回锁定用户时尝试对服务器进行的无效登录的累计次数
Registered	如果已取消注册该对象表示的 Mbean，则返回“False”
LockedUsersCurrentCount	返回服务器上当前锁定的用户数
CachingDisabled	禁用代理中的缓存的专用属性
InvalidLoginAttemptsTotalCount	返回对服务器进行的无效登录尝试的累计次数
UserLockoutTotalCount	返回在服务器上进行的用户锁定的累计次数



8.4 负载压力测试实施

8.4.1 负载压力测试实施步骤

我们将负载压力测试实施步骤概括如下：

测试计划→测试需求分析→测试案例制定→测试环境、工具、数据准备→测试脚本录制、编写与调试→场景制定→测试执行→获取测试结果→结果评估与测试报告。

下面将详细论述各部分的内容。

8.4.2 测试计划

制定一个全面的测试计划是负载测试成功的关键。定义明确的测试计划将确保制定的方案能完成负载测试目标。这部分内容描述负载测试计划过程，包括分析应用程序、定义测试目标、计划方案实施、检查测试目标。在任何类型的系统测试中，制定完善的测试计划是成功完成测试的基础。负载压力测试计划有助于：

① 构建能够精确地模拟工作环境的测试方案。负载测试指在典型的工作条件下测试应用程序，并检测系统的性能、可靠性和容量等。

② 了解测试需要的资源。应用程序测试需要硬件、软件和人力资源。开始测试之前，应了解哪些资源可用并确定如何有效地使用这些资源。

③ 以可度量的指标定义测试成功条件。明确的测试目标和标准有助于确保测试成功。仅定义模糊的目标（如检测重负载情况下的服务器响应时间）是不够的。明确的成功条件应类似于“50 个客户能够同时查看他们的账户余额，并且服务器响应时间不超过 1 分钟”。

下面详细论述负载压力测试计划过程的 4 个步骤。

1．分析应用程序

负载测试计划的第一步是分析应用程序。应该对硬件和软件组件、系统配置以及典型的使用模型有一个透彻的了解。应用程序分析可以确保使用的测试环境能够在测试中精确地反映应用程序的环境和配置。

（1）确定系统组件

绘制一份应用程序结构示意图。如果可能，从现有文档中提取一份示意图。如果要测试的应用程序是一个较大的网络系统的一部分，应该确定要测试的系统组件。确保该示意图包括了所有的系统组件，例如客户机、网络、中间件和服务器等。

如图 8-16 所示说明了一个由许多 Web 用户访问的联机银行系统。各 Web 用户连接

到同一数据库以转移现金和支票余额。客户使用不同的浏览器，通过 Web 方式连接到数据库服务器。

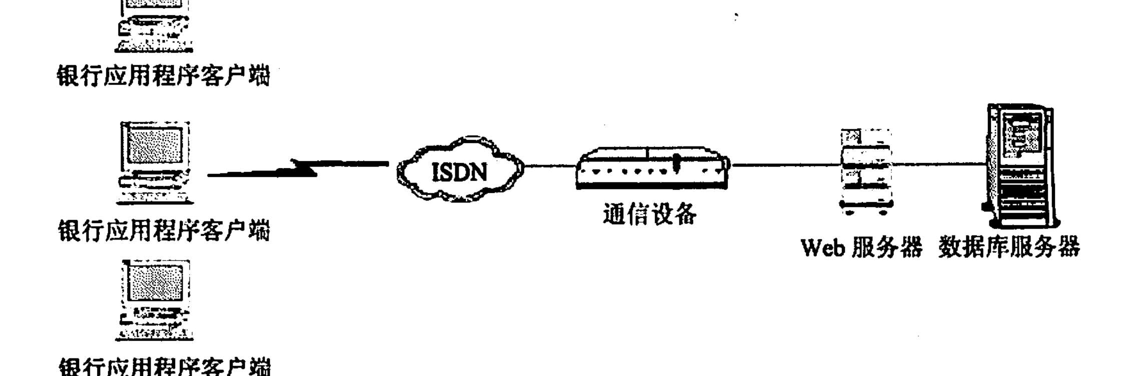


图 8-16　联机银行系统应用布署

（2）描述系统配置

增加更多详细信息以完善示意图。描述各系统组件的配置。应当掌握以下信息：

- 连接到系统的用户数；
- 应用程序客户端计算机的配置情况（硬件、内存、操作系统、软件、开发工具等）；
- 使用的数据库和 Web 服务器的类型（硬件、数据库类型、操作系统、文件服务器等）；
- 服务器与应用程序客户端之间的通信方式；
- 前端客户端与后端服务器之间的中间件配置和应用程序服务器；
- 可能影响响应时间的其他网络组件（调制解调器等）；
- 通信设备的吞吐量以及每个设备可以处理的并发用户数。

例如，在如图 8-16 所示的示意图中，多个应用程序客户端在访问系统。客户端的配置如表 8-16 所示。

表 8-16　客户端配置内容

前端客户端配置	
连接到系统的应用程序客户端的数量	50 个并发应用程序客户端
硬件/内存	586/32MB
操作系统及其版本	Windows NT 4.0
客户端浏览器	Internet Explorer 4.0



（3）分析使用模型

定义系统的典型使用方式，并确定需要重点测试的功能。考虑哪些用户使用系统、每种类型用户的数量，以及每个用户的典型任务。此外，还应考虑任何可能影响系统响应时间的后台负载。

例如，假设每天上午有 200 名员工登录记账系统，并且该办公室网络有固定的后台负载：50 名用户执行各种字处理和打印任务。可以创建一个 200 个虚拟用户登录访问记账数据库的方案，并检测服务器的响应时间。要了解后台负载对响应时间的影响，可以在运行方案的网络中再模拟员工执行字处理和打印活动的负载。

（4）任务分布

除定义常规用户任务外，还应该查看这些任务的分布情况。例如，假设银行用户使用一个中央数据库为跨越多个州和时区的客户提供服务。250 个应用程序客户端分布在两个不同的时区，全都连接到同一个 Web 服务器中。其中 150 个在芝加哥，另外 100 个在底特律。每个客户端从上午 9 点开始工作，但由于处于不同的时区，因此在任何特定时间内都不会有超过 150 个的用户同时登录。可以分析任务分布，以确定数据库活动峰值期的发生时间，以及负载峰值期间的典型活动。

2. 定义测试目标

开始测试之前，应精确地定义想要实现的目标。

（1）以可度量的指标制定目标

确定了负载测试的一般性目标后，应该以可度量指标制定更具针对性的目标。为了提供评估基准，应精确地确定、区分可接受和不可接受测试结果的标准。

例如：

- 一般性目标产品评估：选择 Web 服务器的硬件。
- 明确目标产品评估：在一台 HP 服务器和一台 NEC 服务器上运行同一个包含 300 个虚拟用户的组。当 300 个用户同时浏览 Web 应用程序页面时，确定哪一种硬件提供更短的响应时间。
- 测试目标

① 度量最终用户的响应时间，完成一个业务流程需要多长时间；

② 定义最优的硬件配置，哪一种硬件配置可以提供最佳性能；

③ 检查可靠性，系统无错误或无故障运行的时间长度或难度；

④ 查看硬件或软件升级对性能或可靠性有何影响；

⑤ 评估新产品，应选择哪些服务器硬件或软件；

⑥ 度量系统容量，在没有显著性能下降的前提下，系统能够处理多大的负载；

⑦ 确定瓶颈，哪些因素会延长响应时间。

（2）确定测试的时间

负载测试应贯穿于产品的整个生命周期。如表 8-17 说明了在产品生命周期的各个阶段有哪些类型的测试与之相关。

表 8-17　产品生命周期与测试类型

计划和设计	开　发	部　署	生　产	升　级
评估新产品	度量响应时间	检查可靠性	度量响应时间	检测硬件或软件升级
度量响应时间	检测最优的硬件配置	度量响应时间	确定瓶颈	度量系统容量
	检测硬件或软件升级	度量系统容量		
	检查可靠性			

3．计划方案实施

下一步是确定如何实现测试目标。

（1）定义性能度量的范围

可以度量应用程序中不同点的响应时间。根据测试目标确定在哪里运行 Vuser（虚拟用户）以及运行哪些 Vuser。

- 度量端到端的响应时间。

可以在前端运行 GUI Vuser（图形用户界面用户）或 RTE Vuser（终端用户）来度量典型用户的响应时间。GUI Vuser 可以将输入提交给客户端应用程序并从该应用程序接收输出，以模拟实际用户；RTE Vuser 则向基于字符的应用程序提交输入，并从该应用程序接收输出，以模拟实际用户。

可以在前端运行 GUI 或 RTE Vuser 来度量跨越整个网络（包括终端仿真器或 GUI 前端、网络和服务器）的响应时间。如图 8-17 所示为端到端的响应时间。

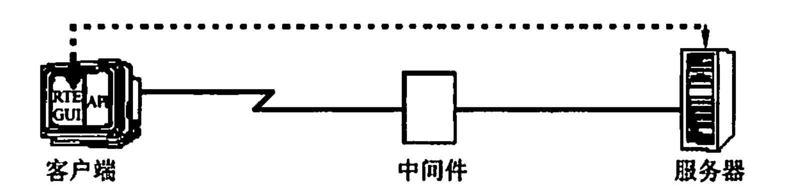


图 8-17　端到端的响应时间

- 度量网络和服务器响应时间。

可以通过在客户机运行 Vuser（非 GUI 或 RTE Vuser）来度量网络和服务器的响应时间（不包括 GUI 前端的响应时间）。Vuser 模拟客户端对服务器的进程调用，但不包括用户界面部分。在客户机运行大量 Vuser 时，可以度量负载对网络和服务器响应时间的



影响。如图 8-18 所示为网络和服务器的响应时间。

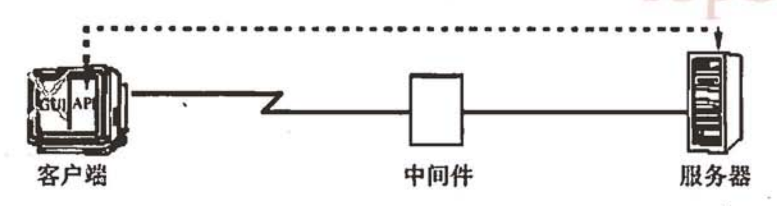


图 8-18 网络和服务器的响应时间

- 度量 GUI 响应时间。

可以通过减去前两个度量值，来确定客户端应用程序界面对响应时间的影响。GUI 响应时间= 端到端响应时间–网络和服务器响应时间。如图 8-19 所示为 GUI 响应时间。

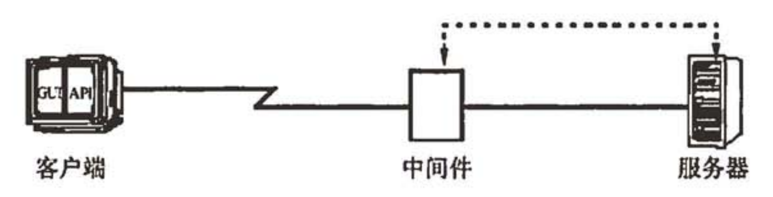


图 8-19 GUI 响应时间

- 度量服务器响应时间。

可以度量服务器响应请求（不跨越整个网络）所花费的时间。通过在与服务器直接相连的计算机上运行 Vuser，可以度量服务器性能。如图 8-20 所示为服务器响应时间。

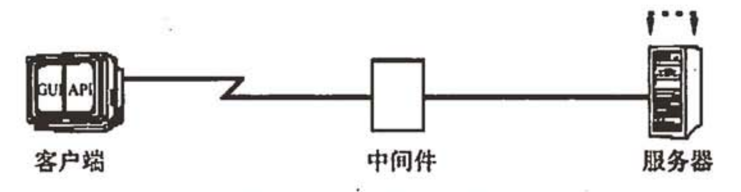


图 8-20 服务器响应时间

- 度量中间件到服务器的响应时间。

如果可以访问中间件及其 API，便可以度量服务器到中间件的响应时间。可以使用中间件 API 创建 Vuser，来度量中间件到服务器的性能。如图 8-21 所示为中间件到服务器响应时间。

（2）定义 Vuser 活动

根据对 Vuser 类型的分析以及它们的典型任务和测试目标来创建 Vuser 脚本。由于 Vuser 模拟典型最终用户的操作，因此 Vuser 脚本应包括典型的最终用户任务。例如，要

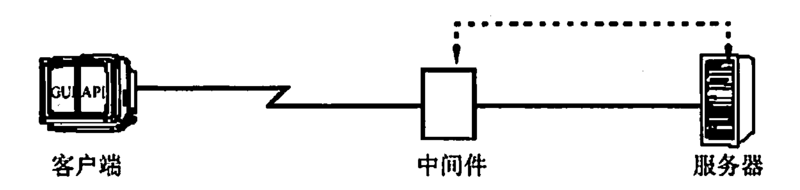


图 8-21　中间件到服务器响应时间

模拟联机银行客户端，应该创建一个执行典型银行任务的 Vuser 脚本。需要浏览经常访问的页面，以转移现金或支票余额。

根据测试目标确定要衡量的任务，并定义这些任务的事务。这些事务度量服务器响应 Vuser 提交的任务所花费的时间（端到端时间）。例如，要查看提供账户余额查询的银行 Web 服务器的响应时间，则应在 Vuser 脚本中为该任务定义一个事务。

此外，可以通过在脚本中使用集合点来模拟峰值期活动。集合点指示多个 Vuser 在同一时刻执行任务。例如，可以定义一个集合点，以模拟 70 个用户同时更新账户信息的情况。

（3）选择 Vuser

确定用于测试的硬件配置之前，应该先确定需要的 Vuser 的数量和类型。要确定运行多少个 Vuser 和哪些类型的 Vuser，请综合考虑测试目标来查看典型的使用模型。以下是一些一般性规则：

- 使用一个或几个 GUI 用户来模拟每一种类型的典型用户连接；
- 使用 RTE Vuser 来模拟终端用户；
- 运行多个非 GUI 或非 RTE Vuser 来生成每个用户类型的其余负载。

例如，假设有五种类型的用户，每种用户执行一个不同的业务流程，如表 8-18 所示。

表 8-18　Vuser 的数量和类型

使 用 模 型	GUI	RTE	其　他
100 个客户服务用户在纽约（LAN 连接）	2	–	98
30 个客户在欧洲（ISDN 拨号连接）	2	–	28
5 个后台批处理进程	–	–	5
150 个客户（终端连接）	–	150	–
6 名管理人员（两个用户使用 486 PC，4 个用户使用 586 PC）	1（486 PC） 1（586 PC）	–	4

（4）选择测试硬件和软件

硬件和软件应该具有强大的性能和足够快的运行速度，以模拟所需数量的虚拟用户。



在确定计算机的数量和正确的配置时，请考虑以下事项。

- 建议在一台单独的计算机上运行测试工具主控台。
- 在一台 Windows 计算机只能运行一个 GUI Vuser；而在一台 UNIX 计算机上则可以运行几个 GUI Vuser。
- GUI Vuser 测试计算机的配置应该尽量与实际用户的计算机配置相同。

关于每个测试组件的硬件要求，请参考表 8-19 和表 8-20。要获得最佳性能，应满足表中所列要求。

表 8-19 测试机硬件与软件要求（Windows 配置要求）

要求	带有联机监视器的 Controller	虚拟 Vuser 生成器	虚拟用户	Analysis 模块
计算机/处理器	Pentium 350 MHz 或更高频率的处理器	Pentium 350 MHz 或更高频率的处理器	Pentium 1GHz 或更高频率的处理器	Pentium 350 MHz 或更高频率的处理器
操作系统	Windows NT®SP 6a 或更高版本 Windows 2000 Windows XP	Windows NT®SP 6a 或更高版本 Windows 2000 Windows XP	Windows NT®SP 6a 或更高版本 Windows 2000 Windows XP HP UX11.x 或更高版本 Solaris 2.6 或更高版本 AIX 4.3.3 或更高版本 Linux Red Hat 6.0 或更高版本	Windows NT®SP 6a 或更高版本 Windows 2000 Windows XP
内存	120 MB 或更大内存	120 MB 或更大内存	对非多线程 Vuser，至少 1 MB RAM； 对多线程 Vuser，至少 512 KB RAM	120 MB 或更大内存
交换空间	总物理内存的两倍	总物理内存的两倍	总物理内存的两倍	总物理内存的两倍
可用硬盘空间	200 MB	200 MB	至少 500 MB	至少 500 MB
浏览器	IE 5.x 或更高版本 Netscape Navigator 4.x、6.x	IE 5.x 或更高版本 Netscape Navigator 4.x、6.x	N/A	IE 5.x 或更高版本 Netscape Navigator 4.x、6.x

注意：对于一个要运行许多事务的长方案，结果文件需要几个 MB 的磁盘空间。负载生成器计算机还需要几个 MB 的磁盘空间来存储临时文件（如果没有 NFS）。有关运行时文件存储的详细信息，请参阅第 10 章“配置方案”。

有关最新的安装要求，请访问

http://www.mercuryinteractive.com/products/loadrunner/technical/

表 8-20 测试机硬件与软件要求（UNIX 配置要求）

要 求	GUI Vuser（每用户）	Vuser（每用户）	Web Vuser（每用户）
内存	4～5 MB，外加客户端应用程序要求	至少 1.5 MB（具体取决于应用程序）	～0.5 MB
交换空间	总物理内存的 4 倍	总物理内存的 4 倍	总物理内存的两倍
磁盘空间	N/A	N/A	N/A
进程数	4	1	1
参与方数量	N/A	N/A	N/A
1 个 CPU 支持的用户数量	30～50 或更多	200～300 或更多	300～400 或更多

注意：对于一个要运行许多事务的长方案，结果文件需要几个 MB 的磁盘空间。负载生成器计算机还需要几个 MB 的磁盘空间来存储临时文件（如果没有 NFS）。有关运行时文件存储的详细信息，请参阅第 10 章“配置方案”。

4. 检查测试目标

测试计划应该基于明确定义的测试目标。下面概述了常规的测试目标。

① 度量最终用户响应时间。

② 定义最优的硬件配置。

③ 检查可靠性。

④ 查看硬件或软件升级。

⑤ 评估新产品。

⑥ 确定瓶颈。

⑦ 度量系统容量。

（1）度量最终用户响应时间

查看用户执行业务流程以及从服务器得到响应所花费的时间。例如，假设想要检测：系统在正常的负载情况下运行时，最终用户能否在 20 秒内得到所有请求的响应。如图 8-22 显示了一个银行应用程序的负载和响应时间度量之间的关系。

（2）定义最优的硬件配置

检测各项系统配置（内存、CPU 速度、缓存、适配器、调制解调器）对性能的影响。了解系统体系结构并测试了应用程序响应时间后，可以度量不同系统配置下的应用程序响应时间，从而确定哪一种设置能够提供理想的性能级别。

例如，可以设置三种不同的服务器配置，并针对各个配置运行相同的测试，以确定性能上的差异。

- 配置 1：200MHz、64MB RAM。
- 配置 2：200MHz、128MB RAM。
- 配置 3：266MHz、128MB RAM。



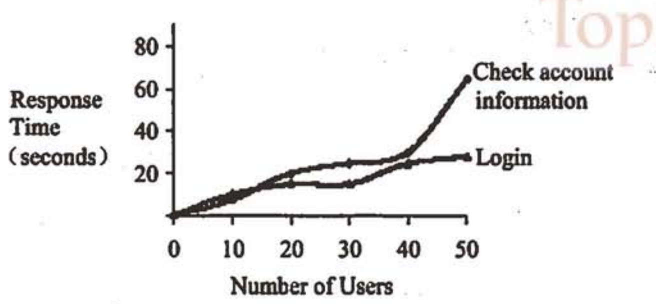


图 8-22 负载和响应时间度量关系

（3）检查可靠性

确定系统在连续的高工作负载下的稳定性级别。可以创建系统负载：强制系统在短时间内处理大量任务，来模拟系统在数周或数月的时间内通常会遇到的活动类型。

（4）查看硬件或软件升级

执行回归测试，以便对新旧版本的硬件或软件进行比较。可以查看软件或硬件升级对响应时间（基准）和可靠性的影响。应用程序回归测试需要查看新版本的效率和可靠性是否与旧版本相同。

（5）评估新产品

可以运行测试，以评估单个产品和子系统在产品生命周期中的计划阶段和设计阶段的表现。例如，可以根据评估测试来选择服务器的硬件或数据库套件。

（6）确定瓶颈

可以运行测试以确定系统的瓶颈，并确定哪些因素导致性能下降，例如，文件锁定、资源争用和网络过载。使用负载压力测试工具，以及网络和计算机监视工具以生成负载，并度量系统中不同点的性能。如图 8-23 所示为系统不同点的性能。

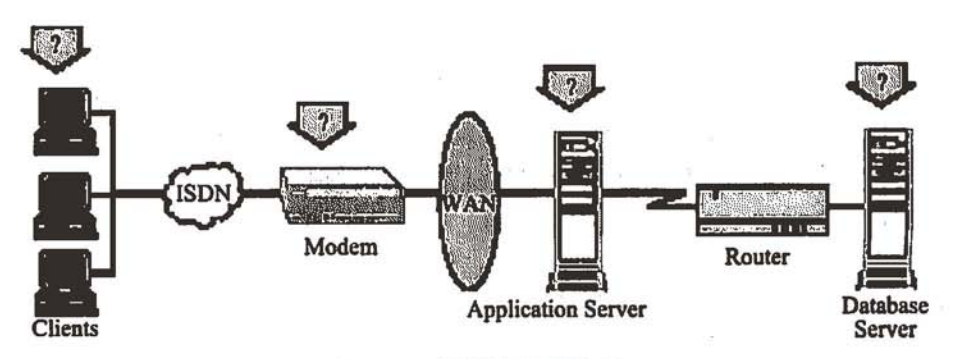


图 8-23 系统不同点的性能

（7）度量系统容量

度量系统容量，并确定系统在不降低性能的前提下能提供多少额外容量。要查看容量，可以查看现有系统中性能与负载间的关系，并确定出现响应时间显著延长的位置。该处通常称为响应时间曲线的“拐点”。确定了当前容量后，便可以确定是否需要增加资源以支持额外的用户。如图 8-24 所示为响应时间与负载关系。

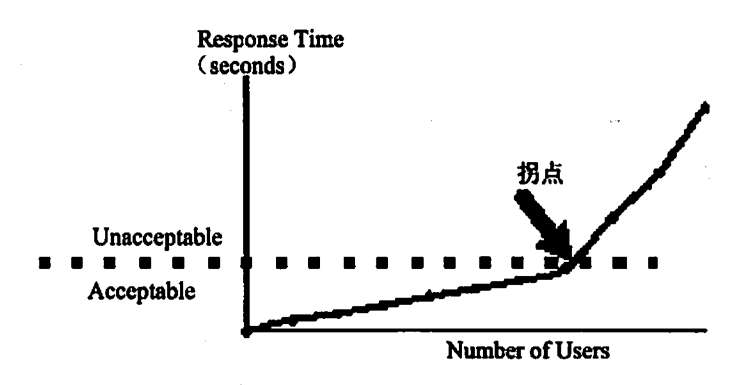


图 8-24　响应时间与负载关系

8.4.3　测试需求分析

负载压力测试需求分析既需要借助于相关的理论知识，又要依靠测试工程师在相关领域的经验积累，下面分别介绍一些理论知识及经验方法。

1．测试需求内容

测试需求是应用需求的衍生，而且测试用例也必须覆盖所有的测试需求，否则，这个测试过程就是不完整的。主要有以下的几个关键点：

- 测试的对象是什么，例如“被测系统中有负载压力需求的功能点包括哪些”；“测试中需要模拟哪些部门用户产生的负载压力”等问题。
- 系统配置如何，例如“预计有多少用户并发访问”；“用户客户端的配置如何”；“使用什么样的数据库”；“服务器怎样和客户端通信”；“网络设备的吞吐能力如何，每个环节承受多少并发用户”等问题。
- 应用系统的使用模式是什么，例如“使用在什么时间达到高峰期”；“用户使用该系统是采用 B/S 运行模式吗”等问题。

我们来针对用户提出的问题，做一个简单的需求回答，如表 8-21 所示。



表 8-21 用户需要与测试目标

测试目标	用户需求
测量对最终用户的响应时间	要花多少时间做完一笔交易
确定最优硬件配置	什么样的配置提供了最好的性能
检查可靠性	系统在无错情况下能承担多大及多长时间的负载
检查软、硬件升级	这些升级对系统性能影响有多大
评估新产品	服务器应该选择哪些硬件与软件
测试系统负载	在没有较大性能衰减的前提下，系统能够承受多大负载
分析系统瓶颈	哪些因素降低了交易响应时间

2．负载压力测试需求分析原理

下面我们介绍 80～20 原理测试强度估算及 UCML 压力需求分析。

（1）80～20 原理测试强度估算

80～20 原理：每个工作日中 80%的业务在 20%的时间内完成。例如，每年业务量集中在 8 个月，每个月 20 个工作日，每个工作日 8 小时即每天 80%的业务在 1.6 小时完成。

举一个例子来看 80～20 原理如何应用与测试需求分析。

去年全年处理业务约 100 万笔，其中，15%的业务处理中，每笔业务需对应用服务器提交 7 次请求；70%的业务处理中，每笔业务需对应用服务器提交 5 次请求；其余 15%的业务处理中，每笔业务需对应用服务器提交 3 次请求。根据以往的统计结果，每年的业务增量为 15%，考虑到今后 3 年业务发展的需要，测试需按现有业务量的两倍进行。

测试强度估算如下。

每年总的请求数为：$(100\times15\%\times7+100\times70\%\times5+100\times15\%\times3)\times2=1000$ 万次/年；

每天请求数为：$1000/160=6.25$ 万次/天；

每秒请求数为：$(62500\times80\%)/(8\times20\%\times3600)=8.68$ 次/秒；

即服务器处理请求的能力应达到 9 次/秒。

（2）UCML（User Community Modeling Language）压力需求分析

UCML™ 是一个符号集合，这些符号可以创建虚拟系统用法模型，以及描述相关参数。当把它应用到负载压力性能测试时，这些符号可用于表示工作量分配、操作流程、重点工作表、矩阵和马尔可夫链等。负载压力性能测试工程师在决定测试中用到什么活动，以及它们发生的频率时，经常用到这些参量。UCML 输出的结论图表可有效地应用于文档，甚至是测试计划、测试设计等，还可作为讨论和数据整理的依据。系统分析人员和用户还可以用它们回顾和验证工作流程和工作量的需求。

UCML™ 给支持负载压力性能测试的复杂数学模型创建一个直观的可视化模型，对于数学模型来说，UCML™ 不是代替品，而是一种补充。现在至少可以证明这一可视化

技术对测试人员、应用用户、开发者、经理和商家来说都是相当直观的。根据我们的经验，它使类似于“此 activity 通用吗”、“为什么 activityB 必须以 activityD 作为先决条件，这是不正确的”，甚至“如何再次阅读重点工作表”这样的重要问题得到了更直观的表现。

没有专门对应于 UCML™ 的画图工具，除了用 SmartDraw 画 UCML™ 模型时，还可以应用 PowerPoint、Microsoft Visio 等软件来画，而且我们可以为 Microsoft Visio 和 SmartDraw 创建符号库。

UCML™ 不是一种工业标准，现在还不能证明是适用于任何类似 IEEE 的工业标准或规范。用户可以应用其中的全部或部分符号，或者按照需要来修改或改进它们。下面我们来看看 UCML™ 的符号。

- 基本符号。

基本符号可以表示系统应用。在许多情况下，一个用法路径可以看作是一个应用会话。事实上，用法路径是 UCML™ 的基础，因为项目组的所有成员早已理解了系统应用。

- 数量环。

数量环是用来回答“多少”、“多大频率”问题的。圆圈里显示的是一个数或者百分比。例如，在模拟精确量的时候用数，不管人数的具体多少；在模拟整个用户群中的一部分时，用百分比表示。如图 8-25 所示为数量环。

1 OR 15%

图 8-25 数量环

- 描述线。

描述线是水平的实线，表示 activity 和用户类型。每个描述线都标注了一个或多个描述内容，来显示用户类型、activity 或运行路径。附在描述语句后面的圆括号里的内容是所有同类用户的百分比，或者是在一定时间段内运行这一 activity、执行这条路径的用户百分比。在一定情况下，用数量环来描述特定用户类型、activity、数量和频率更直观。通常，当有多项标注的 activity 出现或者同一条线上出现相同的数量或频率时，使用数量环，当数量或频率附属于一个特定 activity 时，使用圆括号。

- 用户类型描述线。

用户类型描述线在模型的最左边，表示将要进入已创建的应用中的用户类型，例如：基本用户、超级用户、管理员。与用户类型相关的百分比显示了所有用这种类型表示的系统中的用户比例。如图 8-26 所示为用户类型描述。



图 8-26 用户类型描述



● activity 类型描述线。

activity 类型描述线表示系统的不同类型用户的 activity 和运行路径。线的上面列出了已完成的用户 activity，或运行路径所查看的界面。由建模人员决定应该将特定 activity 的每个步骤列出还是简单地给 activity 命名。记录的关键是每个水平线表示毫无转折的到达该 activity 的直线路径。这条线可以分出支路也可以合并，表示通过该系统对行动路线的选择。选择特定运行路径的用户的百分比显示在路径始端的数量环里。在模型中任何一个指定点，所有运行路径的用户之和都是 100%，与访问站点的人数相同。如图 8-27 所示为 activity 类型描述线。

Search Titles (15%)

图 8-27 activity 类型描述线

● 同步点。

垂直的虚线表示模型中的同步节点。如果同步节点已命名，就将它的名字写在线的上方。同步节点用于描述合并。有两种可以用同步节点符号表示的合并，即时间上的合并和运行点的合并。如图 8-28 所示为同步点。

Sync Point

图 8-28 同步点

时间上的合并意味着在进程开始之前，先到达同步节点的建模用户必须等待所有其他建模用户都到达此节点。在建立信息系统时，这是非常有帮助的。

运行点上的合并意味着，所有建模用户都运行通过了应用中的同一位置，但不一定在同一时间。网站的注册界面就是一个常见的例子。在这个例子中，所有用户在被允许访问其他界面之前，必须在同一网页上填写各自的资格鉴定。在开发脚本时鉴别这些运行同步节点是很有帮助的，因为它们可以为脚本、草稿、功能、程序等充当普通的开始/关闭点。

我们发现，在名称后用圆括号插入同步节点类型的注释很有帮助，例如：“同步点 1 (time)”。

● 选择框。

系统运行时，常常要求用户进行一些不会改变其全部行为的选择或输入数据。模型中，用位于运行路径下方的一个虚线框来表示这些选项。如图 8-29 所示。

选择框需要标注一个词或断语，来描述那些随着用户的改变而变化的选择和数据。特定数据将会记录在模型内的电子数据表中。

Order Book (15%)

Pick Title

图 8-29 选择框

● 条件。

模型中，条件是用户根据结果来改变其运行路径的节点。下面的例子里，假设用户要购买一本书。如果用户查询后，发现它有库存，用户就按照“Yes”路径购买此书。但是，如果此书无库存，用户就按照“No”路径取消此过程。如图 8-30 所示为条件。

- 循环。

在运行路径上，虚线的半圆弧表示循环，半圆弧所环绕的活动将重复执行。需要反复的次数或需要重复此行为的用户百分数(不是模型表示的所有用户)，标注在数量环里。如图 8-31 所示。

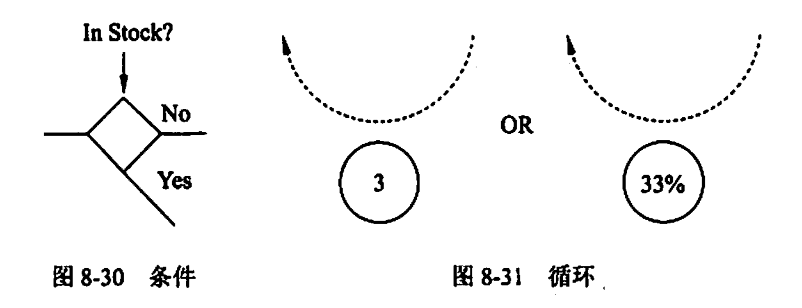


图 8-30　条件　　图 8-31　循环

- 跳出路径。

除了运行路径中断的节点以外，构筑用户的跳出是非常重要的。指向数量环的下划线表示路径在具体某点处，将要跳出系统的用户百分比。记住，所有进入系统的用户都应显示退出。跳出系统的类型应记录在标签上，例如“Log Out”（如果用户基于首选方式退出系统）或“Abandon”（如果用户没有基于首选方式退出系统）。如图 8-32 所示为跳出路径单方式。

下面的例子显示了从运行路径跳出的多种方式。其中，50%的用户放弃了系统，40%的用户正在退出系统，因此他们是以首选方式跳出系统的（另外的 10%从另一个运行路径跳出，没有给出表示）。如图 8-33 所示为跳出路径多种方式。

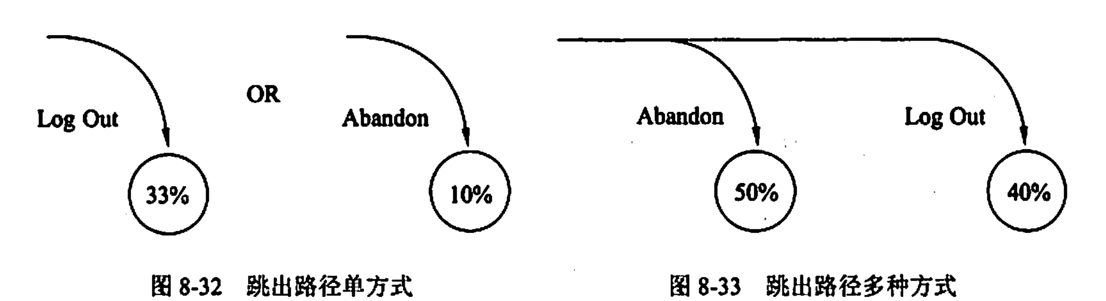


图 8-32　跳出路径单方式　　图 8-33　跳出路径多种方式

- 分支。

大多数应用是不局限于直线路径的。开始执行同一 activity 的一组用户，可能只是为了以后分支到不同的运行路径中。如果我们把 activity 线想象成汽车运行的马路，那么分支就表示由司机进行方向选择的十字路口。下面的例子显示了一个分支，可以在这个

分支上选择两条路径中的一条离开主页。分支上用户百分数的和必须等于引起分支的界面上的用户百分比。这个例子里，主页有100%的用户，而分支上的百分比之和也是100%。如图8-34所示为分支一。

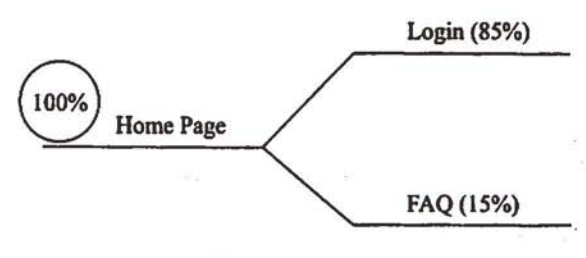


图 8-34 分支一

一个独立的描述线可以有多个分支，如图8-35所示。

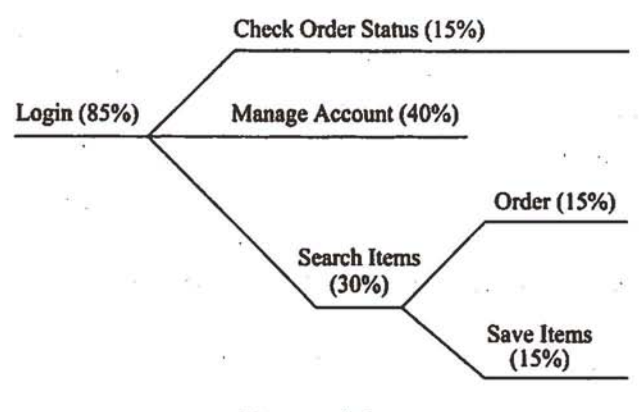


图 8-35 分支二

- 合并。

带有分支的运行路径常遇到将不同分支合并到一起的情况。仍以交通类推，这就相当于来自不同道路的汽车行驶到同一条马路。下面的例子显示了两个不同路径合并到 Update Billing Information 这一 activity。同理，合并前的用户的总百分比必须等于合并后的总用户百分比。如图8-36所示为分支合并。

另外一个常应用的合并是为了显示不同类型的用户在同一点进入了软件，如图8-37所示。这里，每个不同类型的用户用不同的颜色表示。这样做是为了在以后的模型中，使对应于特定用户组的行为能够通过颜色区分。每个用户类型后面都用圆括号标注了它的缩写，这些缩写在以后的模型中也可以应用。这对那种必须由黑白表示的模型特别有利。

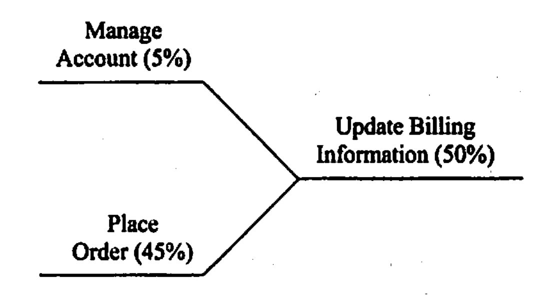


图 8-36　分支合并

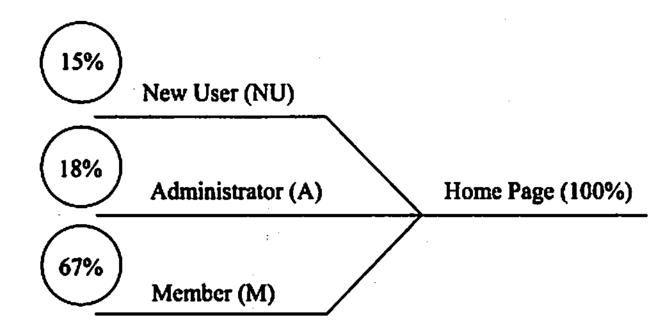


图 8-37　不同类型用户分支合并

我们可以将符号组合为 User Community 模型。

基本符号的组合使 UCML™ 语言更加强大。通过组合符号，可以在一个能形成整个 Community 的虚拟模型的系统中，表示所有可能的用户运行路径。再以交通类比，将模型中的线想象成马路、每个建模用户想象成在这些路上行驶的汽车。把两条或更多条线相遇的点看作十字路口，将同步点看作是交通信号，将跳出看作是驶出轨道。数量环显示出马路的运行载重，因此创建了一个用户如何穿过软件的“地图”。

举个例子，我们将为在线书店建立一个 UCML™ 图表。假设这是一个新的应用，所以我们没有可以分析利用的现存数据。经过一系列的采访，我们收集了下面的信息。它们是关于用户进入网站的 activity 和他们预期的相关量。

- 四种用户类型：新用户（20%），会员（70%），管理员（4%），商家（6%）。
- 所有用户都从主页进入。
- 新用户和会员可引导以下 activity：

① 通过题目、作者、关键字查询书目。

② 向购物车添加一本或更多书。

③ 保留购物车以待结算。

- 新用户可以开设账户（开设账户后就变成了会员）。
- 会员可以选择以下 activity：进入系统、升级账户、项目结算、检查结算状态。
- 管理员和商家必须从主页进入系统，再从管理员界面开始。
- 管理员可以选择以下 activity：添加新书、检查结算状态、升级结算状态、取消命令。
- 商家可执行以下报告：库存、上周销售额、上月销售额。

如图 8-38 所示为在线书店 UCML 图表。

图 8-38　在线书店 UCML 图表

很明显，如图 8-38 所示包含了访问总结中所未提到的信息，这些信息是为最初草案而估计的。仔细观察还会发现，不是所有来自于访问总结的信息都囊括进来了。现在图标可以反馈到被采访者处了，以便他们确认或改正运行路径和用户量、添加或删除 activity。

3．需求分析方法

（1）任务分布图方法

使用任务分布图方法应关注下面两点：

① 有哪些交易任务。

② 在一天的某些特定时刻系统都有哪些主要操作。

举例如图 8-39 所示，可以得到：

- 12:00，交易混合程度最高；
- 10:00～12:00 是登录高峰期；
- 数据更新业务的并发用户数最大为 90，等信息。

并发用户数

案例	2	4	6	8	10	12	14	16	18	20	22	24
登录					22	25	21					
记账						10	15	12		21		
创建记录							18	11	12	85	50	
数据更新							90	75	46	30		
查询						50	30	20	14			
批处理	20	25	15									
生成报表	50	60										40
系统备份	11	8	12									

x

测试执行周期

图 8-39 任务分布图

（2）交易混合图方法

使用交易混合图方法应关注下面三点：

- 系统日常业务主要有哪些操作，高峰期主要有哪些操作。
- 数据库操作有多少。
- 如果任务失败，那么商业风险有多少。

选择重点交易的指标为高吞吐量、高数据库 I/O、高商业风险。举例如表 8-22 所示，“登录”、“生成订单”以及“发货”为负载压力测试重点。

表 8-22 交易混合表

交易名称	日常业务/hr	高峰期业务/hr	Web 服务器负载	数据库服务器负载	风险
登录	70	210	高	低	大
开一个新账号	10	15	中等	中等	小
生成订单	130	180	中等	中等	中
更新订单	20	30	中等	中等	大
发货	40	90	中等	高	大



（3）用户概况图方法

使用用户概况图方法应关注下面两点：

- 哪些任务是每个用户都要执行的；
- 针对每个用户，不同任务的比例如何。

如表 8-23 所示为任务频率表，负载压力测试需要模拟不同用户角色压力。

表 8-23　任务频率表

	定票部门（170）	飞行部门（50）	经理（30）
输入订单	100	25	
更新订单	50	10	
计算飞行里程		70	5
计算销售			8

8.4.4　测试案例制定

测试案例制定包括测试策略、测试案例以及测试内容。

1．测试策略

测试策略一般包括对比测试环境和真实业务测试环境，真实业务操作环境又可能涉及局域网测试环境和机房测试环境等。

2．测试案例

如表 8-24 所示为一个局域网测试案例说明表。

表 8-24　局域网测试案例

案例名称		并发用户数	网络环境（带宽）	数据量	备注说明
制度文档	信息上传	50、100	100M 局域网 56Kbps Modem	50 用户并发，上传 50 条记录； 100 用户并发，上传 100 条记录	只上传信息，不带附件
	文件上传下载	50、100		50 用户并发，上传 50 条记录； 100 用户并发，上传 100 条记录	信息和附件都上传（附件大小为 200k）
项目管理		50、100		50 用户并发，新增 50 条记录； 100 用户并发，新增 100 条记录	
工作记事		50、100		50 用户并发，新增 50 条记录； 100 用户并发，新增 100 条记录	
机房注：记录操作前后数据库记录数目，每个虚拟用户循环执行 3 次案例交易。					

3．测试内容

测试内容一般包括并发性能测试、疲劳强度测试、大数据量测试和系统资源监控等。

8.4.5　测试环境、工具、数据准备

1．测试环境准备

测试环境直接影响测试效果，所有的测试结果都是在一定软硬件环境约束下的结果，测试环境不同，测试结果可能会有所不同，特别对于压力负载测试更是如此，因为压力负载测试结果往往是一组和时间有关的值，因此对负载压力测试环境的准备就显得特别的重要。

（1）测试环境的基本原则

- 要满足软件运行的最低要求，不一定选择将要部署的环境；
- 选用与被测系统相一致的操作系统和软件平台；
- 营造相对独立的测试环境；
- 无毒的环境。

（2）负载压力测试的测试环境

负载压力测试环境准备过程中可以参考测试环境的基本准则，但是又要考虑负载压力测试的特殊性和负载压力测试目的。负载压力测试一般强调“真实”应用环境下的性能表现，从而实现性能评估、故障定位以及性能优化的目的，因此在负载压力测试环境的准备中要注意以下几点：

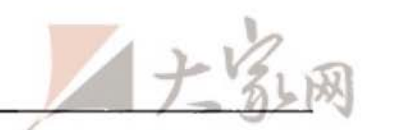
- 如果是完全真实的应用运行环境，要尽可能降低测试对现有业务的影响；
- 如果是建立近似的真实环境，要首先达到服务器、数据库以及中间件的真实，并且要具备一定的数据量，客户端可以次要考虑；
- 必须考虑测试工具的硬件和软件配置要求；
- 配置与业务相关联的测试环境需求；
- 测试环境中应包括对交互操作的支持；
- 测试环境中应该包括安装、备份及恢复过程。

（3）测试环境配置

- 操作系统的版本（包括各种服务、安装及修改补丁）；
- 网络软件的版本；
- 传输协议；
- 服务器及工作站机器；
- 测试工具配置。

（4）良好的测试环境标准

- 保证达到测试执行的技术需求；
- 保证得到稳定的、可重复的、正确的测试结果。



大多数情况下我们强调真实环境下检测系统性能，在实施过程中大家认为这样做会遇到很多阻力，比如真实环境下，不允许负载压力测试为系统带来大量的垃圾数据；测试数据与真实业务数据混在一起无法控制测试结果；负载压力测试如果使服务器宕机，会给系统带来巨大损失等。那么我们应该如何理解“真实环境下检测系统性能”呢？

在负载压力测试中我们强调的“真实环境”是指后台服务器与客户端应用要与实际真实应用环境保持一致，同时，这里也包括了与业务有关的软硬件配置环境和数据量环境等，可以看出我们将网络环境排除在外。原因是网络环境缓解了客户端对服务器所造成的并发负载压力，网络规模越大、网络类型越多、网络拓扑越复杂、网络流量越纷繁交织对客户端的并发负载压力缓解程度越大。

2．测试工具准备

这里主要讨论选择测试工具的原则，介绍一些主流负载压力测试工具，同时也涉及到测试工具的缺陷等内容。

（1）负载压力测试工具选择

古人云“工欲善其事，必先利其器”，进行负载压力性能测试首先应该选择一个合适的测试工具，一个测试工具能否满足测试需求、能否达到令人满意的测试结果，是选择测试工具要考虑的最基本的问题。更进一步可以考虑一些细节问题，比如该工具对于处理扩展的交互（例如一个请求取决于上一个请求的结果）如何；对于处理 cookies（cookies 对于许多面向会话的 J2EE 系统是必不可少的）如何；如果 J2EE 应用程序客户机需要处理一些 JavaScript，以进入下一次通信会如何处理；在收集了响应时间数据后，如何对它进行分析；对 CPU 时间、网络使用、堆大小、分页活动或者数据库活动如何监控等都是选择一个测试工具需要考虑的具体问题。一些高端工具与基本工具往往在一些细节、适用范围以及易用性方面存在差别。比如，顶级的负载测试工具可以模拟多个浏览器，与大多数应用服务器集成，收集多个服务器主机的性能数据（包括操作系统、JVM 和数据库统计数字），生成可以在以后用高级的分析工具分析的数据集。

当然，测试工具的选择首先应该看是否能够满足基本测试需求，该工具必须可以模拟应用程序客户机，如果应用程序使用一些不常见的浏览器功能组合或者其他非标准客户机技术，那么就排除了相当一部分候选者。具备了基本功能后，可以考虑工具的生产率。一般来说，包含的分析工具越多，可以记录的性能数据类型越多，可以达到的生产率就越高，价格也就越高。不过有些低端负载测试程序是免费的，在预算有限的情况下，“免费”的意义是不言自明的。

具体来说在选择测试工具时可以考虑以下几点。

- 模拟客户机。

首要要求一定是负载测试程序能够处理应用程序所使用的功能和协议。

- 运行多个模拟的客户机。

这是负载测试程序最基本、且最重要的功能，它有助于确定哪些是负载测试程序以及哪些不是。

- 脚本化执行并能编辑脚本。

如果不能编辑客户机与服务器之间交互的脚本，那么就不能处理除最简单的客户机之外的任何东西。编辑脚本的能力是最基本的，且小的改变不应该要求重新生成脚本。

- 支持会话。

如果不支持会话或者 cookies，就不算是真正的负载压力测试工具，并且不能对大多数 J2EE 应用程序进行负载测试。

- 可配置的用户数量。

测试程序应该可以让您指定每个脚本由多少个模拟用户运行，包括让您随时间改变模拟用户的数量，因为许多负载测试可以做到从小的用户数量开始，并慢慢增加到更多的用户数量。

- 报告成功、错误和失败。

每一个脚本都必须定义一个方法来识别成功的交互以及失败和错误模式（错误一般不会有页面返回，而失败可能在页面上得到错误的数据）。

- 页面显示。

如果测试工具可以检查一些发送给模拟用户的页面，这会很有用，这样可以做到心中有数，以确保测试工作是正确进行的。

- 导出结果。

可以用不同的工具来分析测试结果，这些工具包括电子表格和可以处理数据的自定义脚本。虽然许多负载测试工具包括大量的分析功能，但是导出数据的能力使您在以任意的方式分析和编辑数据方面具有更大的灵活性。

- 考虑时间。

真实世界的用户不会在收到一页后立即请求另一页，一般在查看一页和下一页之间会有延迟。考虑时间这个标准术语表示在脚本中加入延迟以更真实地模拟用户行为。大多数负载测试程序支持根据统计分布随机生成考虑时间。

- 客户机从列表中选择数据。

用户一般不会使用同样的一组数据，每位用户通常与服务器进行不同的交互。模拟用户应该也这样做，如果在交互的关键点，脚本可以从一组数据中选择数据，则可以更容易地让您的模拟用户表现出使用不同数据的行为。

- 从手工执行的会话记录脚本。

相对于编写脚本，用浏览器手工运行会话并记录这个会话，然后再编辑会容易

得多。

- JavaScript。

一些应用程序大量使用 JavaScript 并且需要模拟客户机支持它。不过，使用客户端 JavaScript 可能会增加对测试系统上系统资源的需求。

- 分析工具。

得到测试数据只是成功的一半，另一半是分析性能数据。因此测试工具提供的分析工具越多，就越有利于从不同的方面进行数据分析。

- 测量服务器端统计数字。

基本负载压力测试工具测量客户机/服务器交互中基于客户机的响应时间。如果同时收集其他统计数据（如 CPU 使用情况和页面错误率）就更好了，有了这些数据，就可以进一步做一些有用的工作，如查看服务器负载上下文中的客户机响应时间和吞吐量统计。

（2）负载压力测试工具的局限性

任何负载压力测试工具都不是完美无缺的，在我们的实际使用中经常碰到一些问题，概括起来主要有以下几点：

- 缺乏功能点的校验；
- 对有些控件支持得不好；
- 不能达到真实模拟负载；
- 脚本的支持不够灵活；
- 报错定位不够详细。

在实际压力测试过程中，我们通常使用多种工具达到测试目的，如何配合使用多种工具，是否能达到最好的"性能价格比"？测试工程师要发挥最大的主观能动性。

（3）主流负载压力测试工具介绍

- QALoad——美国 Compuware（康博）公司。

特点：

① 测试接口。DB2，DCOM，ODBC，ORACLE，NETLoad，Corba，QARun，SAP，SQLServer，Sybase，Telnet，TUXEDO，UNIFACE，WinSock，WWW。

② 预测系统性能。当应用升级或者新应用部署时，负载测试能帮助确定系统是否能按计划处理用户负载。QALoad 并不需调用最终用户及其设备，它能够仿真数以千计的用户进行商业交易。通过 QALoad，用户可以预知业务量接近投产后的真实水平时端对端的响应时间，以便满足投产后的服务水平要求。

③ 通过重复测试寻找瓶颈问题。QALoad 录制/回放能力提供了一种可重复的方法来验证负载下的应用性能，可以很容易地模拟数千个用户，并执行和运行测试。利用 QALoad 反复测试可以充分地测试与容量相关的问题，快速确认性能瓶颈并进行优化和

调整。

④ 从控制中心管理全局负载测试。QALoad Conductor 工具为定义、管理和执行负载测试提供了一个中心控制点。Conductor 通过执行测试脚本管理无数的虚拟用户。Conductor 可以自动识别网络中可进行负载测试的机器，并在这些机器之间自动分布工作量以避免网段超载。从 Conductor 自动启动和配置远程用户，跨国机构可以进行全球负载测试。在测试过程中，Conductor 还可以在负载测试期间收集有关性能和时间的统计数据。

⑤ 验证应用的可扩展性。出于高可扩展性的设计考虑，QALoad 包括了远程存储虚拟用户响应时间，并在测试结束后或其他特定时间下载这些资料的功能。这种方法可以增加测试能力，减少进行大型负载测试时的网络资源耗费。QALoad 采用轮询法采集响应时间，在无需影响测试或增加测试投资的条件下，就可了解测试中究竟出现了什么情况。

在利用 QALoad 进行测试时，可以有选择地改变硬件或软件的配置，并改变测试步调和负载量。QALoad 系统的 NetLoad 模块帮助建立所需的额外网络流量来模拟真实的产品负载。借助于“NetLoad”，可以在大环境下分配虚拟用户，更好地规划在投产环境中如何让这些应用更好地工作。

QALoad 引入了“flash”为电子商务应用软件作容量测试。flash 测试允许在测试期间增加大量的虚拟用户，来确定压力对底层部件和基础结构的影响。flash 测试也可以证明应用投产后其响应时间将不会降低至服务级以下。

⑥ 快速创建仿真的负载测试。准确仿真复杂业务的进行，对于预测电子商务应用软件的功能至关重要。运用 QALoad，可以迅速创造出一些实际的安装测试方案，而不需要手工编写脚本或有关应用中间软件的详细知识和协议。

对结合了多种传输协议的应用软件进行负载测试是一个巨大的挑战。为了准确仿真这些应用软件产生的流量，QALoad 可以捕获多种协议并在同一测试过程中执行它们。基于浏览器的应用经常打开与服务器的多个连接以缩短网页下载时间，导致服务器的额外流量。QALoad 能准确地仿真出浏览器与服务器之间的交互，包括多重连接，使负载更加准确。

另外，这些应用软件常常包含一些在测试方案中必须申明的动态信息。运用 QALoad 的 ActiveData 特征，可以定义脚本参数，以帮助确保应用测试脚本的成功执行。

例如，对一些安全或非安全的 Web 应用软件，ActiveData 自动转化为动态信息，如 cookies，包括动态 Active Serve Page cookies、动态 URL 名称、服务器导向、动态框架或者其他时期的特定信息。ActiveData for Web 有助于保证测试脚本与 Web 应用在测试过程中保持同步。

ActiveData 每次自动查找和提出需要的变更建议，使测试脚本正确执行。在脚本执行过程中，脚本中的信息可以被产生的正确信息自动替代。



⑦ 性能价格比较高。

这些特点应该是负载压力测试工具普遍具备的特点，在后面的负载压力工具介绍中就不再赘述，而是介绍一些除了这些之外的特点。

测试结果报告如图 8-40 所示。

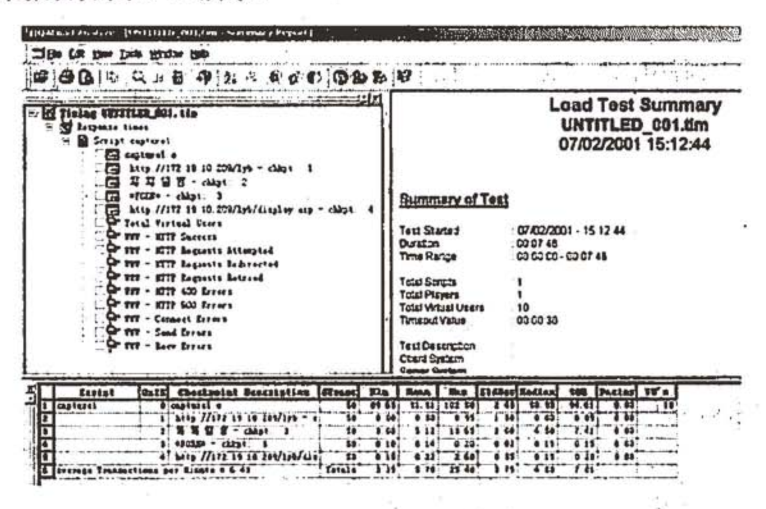


图 8-40　QALoad 测试报告

测试结果分析如图 8-41 所示，1－2－3 表示业务组 A，4－5－6 表示业务组 B，7－8－9 表示业务组 C，10－11－12 表示业务组 D，在各个业务组中序号从小到大表示不同的并发用户。

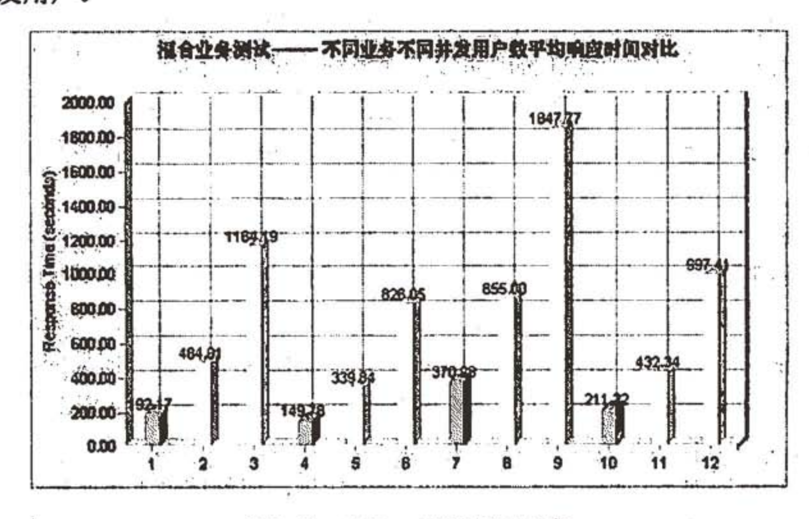


图 8-41　QALoad 测试结果分析

- LoadRunner——美国 Mercury Interactive 公司。

特点：

① 测试接口：支持的协议多且个别协议支持的版本较高；

② 负载压力测试方案设置灵活；

③ 丰富的资源监控，资源监控计数器，如图 8-42 所示；

④ 报告可以导出到 Word、Excel 以及 HTML 格式。

LoadRunner's Real-time Performance Monitors
LoadRunner's real-time performance monitors help isolate performance problems within your system.

Os Monitors:
- Windows NT, 2000
- UNIX
- Linux

Network Monitors:
- SNMP
- Network Delay

Firewall Monitors:
- CheckPoint

Middleware Monitors
- BEA Tuxedo
- Enterprise Java Beans

Web Server Monitors:
- Microsoft IIS
- IPlanet (NES)
- Apache

Web Application Server Monitors:
- Broad Vision One-To-One
- Allaire ColdFusion
- SIlverStream
- BEA WebLogic Server
- IBM WebSphere
- Microsoft ASP
- Ariba Buyer
- ATG Dynamo
- IPlanet (NAS)

Streaming Monitors:
- RealNetworks
- Microsoft Windows Media

Database Monitors:
- Oracle
- Microsoft SQL Server
- IBM DB2

图 8-42　LoadRunner 资源监控计数器

- Benchmark Factory——美国 Quest 软件公司。

特点：

① 可以测试服务器集群的性能；

② 可以实施基准测试；

③ 可以生成高级脚本，例如脚本中数据池的生成可以采用随机数。

- WAS——美国 Microsoft 公司。

这是一个 Microsoft 提供的免费的 Web 负载压力测试工具，应用比较广泛，下面做



一个较详细的介绍。

WAS 可以通过一台或者多台客户机模拟大量用户的活动。WAS 支持身份验证、加密和 Cookies，也能够模拟各种浏览器类型和 Modem 速度，它的功能和性能可以与数万美元的产品相媲美。

要对网站进行负载测试首先必须创建 WAS 脚本模拟用户活动。我们可以用下面四种方法之一创建脚本：① 通过记录浏览器的活动；② 通过导入 IIS 日志；③ 通过把 WAS 指向 Web 网站的内容；④ 手工制作。

制作 WAS 脚本是相当简单的，不过要制作出模拟真实用户活动的脚本就有些复杂。如果你已经有一个运行的 Web 网站，可以使用 Web 服务器的日志来确定 Web 网站上的用户点击分布。如果你的应用还没有开始运行，那么只好根据经验作一些猜测了。

客户端交易测试指标主要有：

① Number of hits：测试间隔内虚拟用户点击页面的总次数；

② Requests per second：每秒客户端的请求次数；

③ Threads：线程数；

④ TTFB Avg：从第一个请求发出到测试工具接收到服务器应答数据的第一个字节之间的平均时间；

⑤ TTLB Avg：从第一个请求发出到测试工具接收到服务器应答数据的最后一个字节之间的平均时间。

准备好测试脚本之后，我们可以调整测试配置，以便观察不同条件下的应用性能。如图 8-43 所示是 WAS 的设置界面。

Stress Level 和 Stress Multiplier 这两项决定了访问服务器的并发连接的数量。如果要模拟的并发连接数量超过 100 个，可以调整 Stress multiplier 或使用多个客户机。在负载测试期间 WAS 将通过 DCOM 与其他客户机协调。有关在测试中使用多个客户机的更多信息，参见http://webtool.rte.microsoft.com/kb/hkb13.htm。

如果网站提供个性化服务，要进行身份验证或使用 Cookies，我们还要为 WAS 提供一个用户目录。WAS 中的用户存储了发送给服务器的密码以及服务器发送给客户端的 Cookies。增加用户数量并不增加 Web 服务器的负载。必须提供足够数量的用户以满足并发连接的要求（Stress Level 乘以 Stress Multiplier）。有关线程、用户和 Cookies 相互作用的更多信息请参见http://webtool.rte.microsoft.com/Threads/WASThreads.htm。

WAS 允许设置 warmup（热身）时间，一般可以设置为 1 分钟。在 warmup 期间 WAS 开始执行脚本，但不收集统计数据。warmup 时间给 MTS、数据库以及磁盘缓冲等一个机会来做准备工作。如果在 warmup 时间内收集统计数据，这些操作的开销将影响性能测试结果。

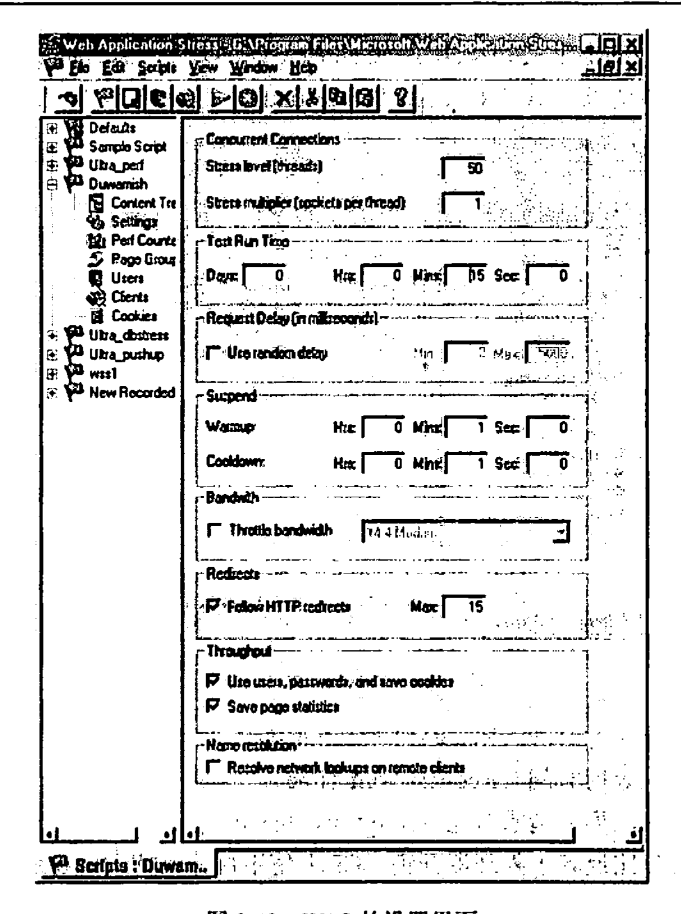


图 8-43　WAS 的设置界面

设置页面提供的另外一个有用的功能是限制带宽（throttle bandwidth）。带宽限制功能能够为测试模拟出 Modem（14.4K，28.8K，56K）、ISDN（64K，128K）以及 T1（1.54M）的速度。使用带宽限制功能可以精确地预测出客户通过拨号网络或其他外部连接访问 Web 服务器所感受的性能。

要理解这些不同的设置对应用的影响，有必要了解如何使用 WAS 收集性能数据。

使用 WAS，从远程 Windows NT 和 Windows 2000 机器获取和分析性能计数器（Performance Counter）是很方便的。加入计数器要用到如图 8-44 所示的 Perf Counters 分支。

在测试中选择哪些计数器显然跟测试目的有关。虽然下面这个清单不可能精确地隔离出性能瓶颈所在，但对一般的 Web 服务器性能测试来说却是一个好的开始。

① 处理器：CPU 使用百分比（% CPU Utilization）；



② 线程：每秒的上下文切换次数（Context Switches Per Second (Total)）；

③ ASP：每秒请求数量（Requests Per Second）；

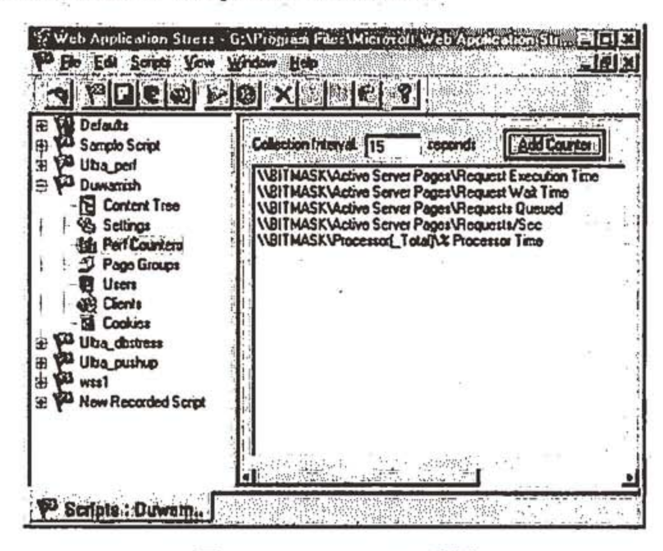


图 8-44　Perf Counters 分析

④ ASP：请求执行时间（Request Execution Time）；

⑤ ASP：请求等待时间（Request Wait Time）；

⑥ ASP：置入队列的请求数量（Requests Queued）。

CPU 使用百分比反映了处理器开销。CPU 使用百分比持续地超过 75%是性能瓶颈存在于处理器的一个明显的迹象。每秒上下文切换次数指示了处理器的工作效率。如果处理器陷于每秒数千次的上下文切换，说明它忙于切换线程而不是处理 ASP 脚本。

每秒的 ASP 请求数量、执行时间以及等待时间在各种测试情形下都是非常重要的监测项目。每秒的请求数量表示每秒内服务器成功处理的 ASP 请求数量。执行时间和等待时间之和显示了反应时间，这是服务器用处理好的页面作应答所需要的时间。

可以绘出随测试中并发用户数量的增加，每秒请求数量和反应时间的变化图。增加并发用户数量时每秒请求数量也会增加。然而，我们最终会达到这样一个点，此时并发用户数量开始“压倒”服务器。如果继续增加并发用户数量，每秒请求数量开始下降，而反应时间则会增加。要搞清楚硬件和软件的能力，找出这个并发用户数量开始“压倒”服务器的临界点非常重要。

置入队列的 ASP 请求数量也是一个重要的指标。如果在测试中这个数量有波动，某个 COM 对象所接收到的请求数量超过了它的处理能力。这可能是因为在应用的中间层使用了一个低效率的组件，或者在 ASP 会话对象中存储了一个单线程的单元组件。

运行 WAS 的客户机 CPU 使用率也有必要监视。如果这些机器上的 CPU 使用率持续地超过 75%，说明客户机没有足够的资源来正确地运行测试，此时应该认为测试结果不可信。在这种情况下，测试客户机的数量必须增加，或者减小测试的 Stress Level。

每次测试运行结束后 WAS 会生成详细的报表，即使测试被提前停止也一样。WAS 报表可以从 View 菜单选择 Reports 查看。下面介绍一下报表中几个重要的部分。

如果这是一个新创建的测试脚本，你应该检查一下报表的 Result Codes 部分。这部分内容包含了请求结果代码、说明以及服务器返回的结果代码的数量。如果这里出现了 404 代码（页面没有找到），说明在脚本中有错误的页面请求。

页面摘要部分提供了页面的名字，接收到第一个字节的平均时间（TTFB），接收到最后一个字节的平均时间（TTLB），以及测试脚本中各个页面的命中次数。TTFB 和 TTLB 这两个值对于计算客户端所看到的服务器性能具有重要意义。TTFB 反映了从发出页面请求到接收到应答数据第一个字节的时间总和（以毫秒计），TTLB 包含了 TTFB，它是客户机接收到页面最后一个字节所需要的累计时间。

报表中还包含了所有性能计数器的信息。这些数据显示了运行时各个项目的测量值，同时还提供了最大值、最小值、平均值等。报表实际提供的信息远远超过了我们这里能够介绍的内容。为了给你一个有关报表所提供信息种类的印象，摘录了一个报表实例如图 8-45 所示。

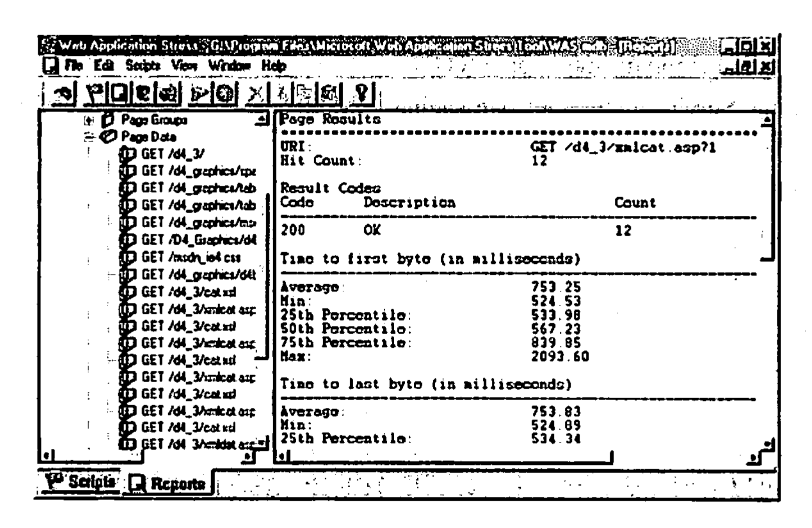


图 8-45　报表实例

- SILK PERFORMER V——美国 Segue 公司。

特点：



① 在工具中融合了功能测试的方法，即内容校验。

② 脚本采用 PASCAL，资源消耗较小，支持一些底层访问。

③ 错误可精确定位。

④ 提供数据池模板，并可定制。

3．测试协议选择

协议这个概念大家并不陌生，测试工具中的“协议”指的是工具提供给我们的测试接口，也可以理解为测试类型。LoadRunner 提供的测试协议比较全面，例如：

- 应用程序解决方案：Citrix。
- Client/Server：MS SQL，ODBC，Oracle（2-tier），DB2 CLI，Sybase Ctlib，Sybase Dblib，Windows Sockets 及 DNS。
- 定制：C templates，Visual Basic templates，Java templates，JavaScript 及 VBScript。
- 分布式组件：COM/DCOM，Corba-Java 及 Rmi –Java。
- E-business：FTP，LDAP，Palm，SOAP，Web（HTTP/HTML），及 the dual Web/Winsocket。
- Enterprise Java Beans：EJB Testing 及 Rmi-Java。
- ERP/CRM：Baan，Oracle NCA，Peoplesoft-Tuxedo，Peoplesoft 8 Web multilingual，SAPGUI，SAP-Web，Siebel（Siebel-DB2CLI，Siebel-MSSQL，Siebel-Web 及 Siebel-Oracle）。
- Legacy：Terminal Emulation（RTE）。
- Mailing Services：Internet Messaging（IMAP），MS Exchange（MAPI），POP3 及 SMTP。
- Middleware：Jacada 及 Tuxedo（6，7）。
- Streaming：MediaPlayer 及 RealPlayer。
- Wireless：i-Mode，VoiceXML 及 WAP。

这里我们要重点讨论的问题是选择测试协议的策略。一个原则性的观点是“客户端与直接压力承受的服务器之间的通信协议是选择测试协议的惟一标准”。例如，有的测试工程师问“我们的系统是 B/S 运行模式，应该选择什么样的测试协议来测”，我们说这是一个无效问题，为什么呢？从这个问题中我们不能获取任何与通信协议有关的信息，B/S 运行模式可以采用 http 协议，也可以采用 TCP/IP，SMTP、FTP 等协议，C/S 运行模式也是这个道理。选择不同的协议决定了测试的成功与失败。理论知识要适用于实践，必须活学活用。再例如，一个使用非常普遍的系统：B/S 运行模式，前端 IE 浏览器，IE 浏览器直接与 Web 服务器通信（可能多台），Web 服务器与后台数据库服务器（可能多台）有数据交互操作，IE 浏览器与 Web 服务器的通信协议采用 http，那么，理所当然我

们选择的测试协议是 http；再灵活一些，如果 IE 浏览器与 Web 服务器的通信不仅采用了 http 协议，而且还有部分业务采用 Winsocket，那么必须选择 Web/Winsocket 双协议；更进一步，有些系统客户端是 C/S 运行模式和 B/S 运行模式的混合，为了达到测试目的，就要选择更多的测试协议。可喜的是 LoadRunner 7.8 版本以后已经能够帮你实现这个愿望了。再来看一种情况：系统的架构是客户端应用程序＋Tuxedo 消息中间件（或者是其他中间件）＋数据库服务器。遇到这种情况，我们一般会选择 Tuxedo 测试协议，能够有这样的定位选择，BEA 公司的产品 Tuxedo 的市场占有率给测试工具厂商带来的压力显而易见。还要跟大家介绍一种普遍认为最"惨"，也最"酷"的方法，就是利用测试工具提供的编程语言自己编写测试脚本，测试工具提供的使用广泛的测试脚本是 CScript、JavaScript 及 VBScript。但提醒大家的是，要做到真实模拟负载，和相关开发人员的交流至关重要。

4．测试数据准备

（1）测试数据概念

实施负载压力测试时，需要运行系统相关业务，这时需要一些数据支持才可运行业务，这部分数据即为初始测试数据。例如 ERP 软件运行前财务账套的准备。

在初始的测试环境中需要输入一些适当的测试数据，目的是识别数据状态并且验证用于测试的测试案例。在正式的测试开始以前对测试案例进行调试，将正式测试开始时的错误降到最低。在测试进行到关键过程领域时，非常有必要进行数据状态的备份。制造初始数据意味着将合适的数据存储下来，需要的时候恢复它，初始数据提供了一个基线用来评估测试执行的结果。

对系统实施负载压力测试的时候，经常会需要准备大数据量、实施独立的测试，或者与并发负载压力相结合的性能测试，这部分数据为业务测试数据。例如飞机订票系统查询订票信息，就需要准备大量的订票记录。又比如测试并发查询业务，那么要求对应的数据库和表中有相当的数据量，以及数据的种类应能覆盖全部业务。

在负载压力测试过程中，为了模拟不同的虚拟用户的真实负载，需要将一部分业务数据参数化，这部分数据为参数化测试数据。例如模拟不同用户登录系统，就需要准备大量用户名及相应密码参数数据。

还需要考虑特殊系统需要的测试数据，模拟真实环境测试，有些软件特别是面向大众的商品化软件，在测试时常常需要考察在真实环境中的表现。如测试杀毒软件的扫描速度时，硬盘上布置的不同类型文件的比例要尽量接近真实环境，这样测试出来的数据才有实际意义。

（2）为什么要准备测试数据

可以概括为：



- 识别数据状态；
- 验证测试案例；
- 初始数据提供了一个基线用来评估测试执行的结果；
- 业务数据提供负载压力背景；
- 脚本中参数数据真实模拟负载。

（3）依靠工具准备测试数据的方法

在 8.2.3 小节，介绍了依靠工具准备测试数据的方法。

5．自己动手编写测试工具

下面通过列举两个实例，来了解自己动手编写测试工具的思路。

（1）案例一：Web 服务器通用性能测试系统的设计与实现

这个系统不仅能够测试静态 HTML 页面的响应时间，而且能够模拟真实运行情况测试动态网页（包括 ASP、PHP、JSP 等）的响应时间，为服务器的性能优化和调整提供数据依据。

为了能够模拟大量用户同时访问 Web 动态页面的情况，必须要解决下面两个问题：

- 测试工具需要模拟出大量用户同时访问 Web 的情况；
- 测试工具需要模拟出单个用户访问 Web 时个性化的请求参数。

本性能测试系统主要由两部分构成：性能测试数据文件和性能测试程序。其中，性能测试数据文件包含着用户访问 Web 的 URL 请求格式和大量用户访问系统的请求数据（例如用户名和密码等）。实际测试中，性能测试程序将开设多个进程模拟大量用户对 Web 的访问，每个进程从性能测试数据文件中随机地读出一组访问数据，然后发起对 Web 服务器的访问，等待 Web 服务器应答。测试结束后，性能测试程序将给出对于所有请求的平均系统响应时间。由此，本性能测试工具能够真实地模拟大量用户同时访问 Web 的情况。

在实际运行的 Web 应用系统中，用户访问动态页面时传递的 query 字符串中的参数是互不相同的。为了逼真地模拟实际情况，性能测试系统应该在一段特定的时间内，对待测页面同时发送多个请求，每个请求的 query 参数互不相同，在发送的同时开始计时，直到收到系统的响应，计时停止，最后对所有请求的响应时间进行分析，就可以得到系统性能的定量估计。系统结构如图 8-46 所示。

实际系统由四部分组成：

- 模板文件。
- 数据文件。
- 性能测试程序。
- 结果处理程序。

模板文件为纯文本文件，包含单个用户访问 Web 发送的 URL，每行格式如下：

GET(POST) http://host:port/path/filename?xxx=@1@&@2@

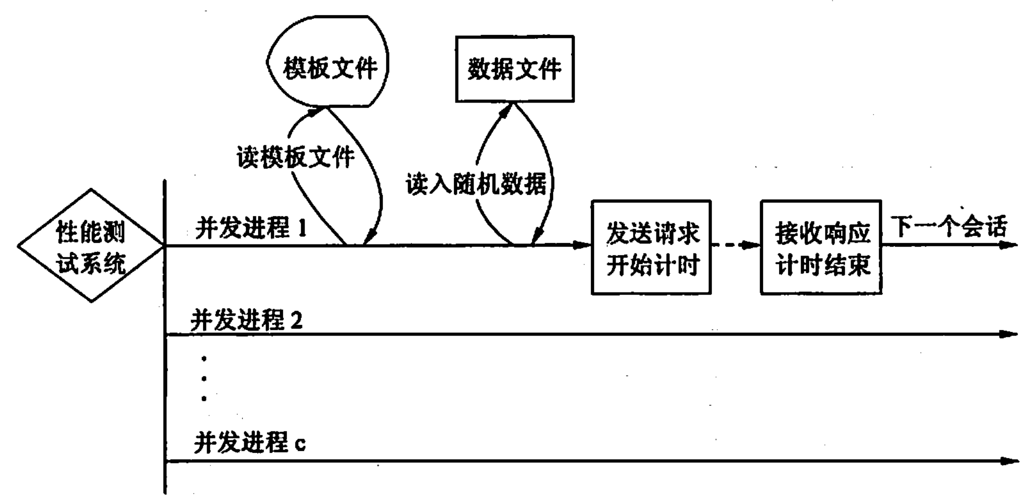


图 8-46　系统结构示意图

其中 GET 或者 POST 表示参数传递的方式，query 字符串中的@是本系统特设字符，表示@及其后面的数字需要被数据文件中对应的参数数据所取代。两个@之间为参数序号，只能为数字，在整个模板文件中相同的参数序号代表相同的参数。

数据文件是用户访问动态页面时传递的 query 参数集合，为纯文本文件，参数数据与模板文件中@号及其后面的数字相对应，之间用空格隔开，每组一行。

测试 Web 应用系统时，性能测试程序将开设 c 个进程，每个进程可以串行地开设 n 个会话，每个会话模拟一个真实用户，按照模板文件中提供的访问 Web 系统的格式，从数据文件中读取一组 query 参数，然后对 Web 系统发起请求；与此同时，程序开始计时，直到收到系统的响应，计时停止，系统统计接收的字节数，将结果写入结果文件，本次会话结束，这个进程开始一个新的会话，如此循环 n 次。测试结束后，结果处理程序对结果文件中每个请求的响应时间进行统计分析，给出系统的综合性能评估。

从上面的分析可以看出，本系统发送请求的并发度是 c，即在测试的时间段内，对 Web 应用系统同时发送的请求有 c 个；本系统发送请求的串行度为 n，一共可以模拟 c×n 个用户对系统的访问。

性能测试程序流程如图 8-47 所示。

如图 8-47 所示，性能测试程序 fork 出 c 个进程，每个进程都开设一个 Socket，通过 Socket 向 Web 服务器发送请求。为了使测试程序能够快速地向服务器发送请求，程序一开始就将数据文件中的所有数据读入内存，数据使用一个二维数组存放。

模板文件中的请求格式在程序开始时读入模板数据结构中，模板数据结构定义如下：

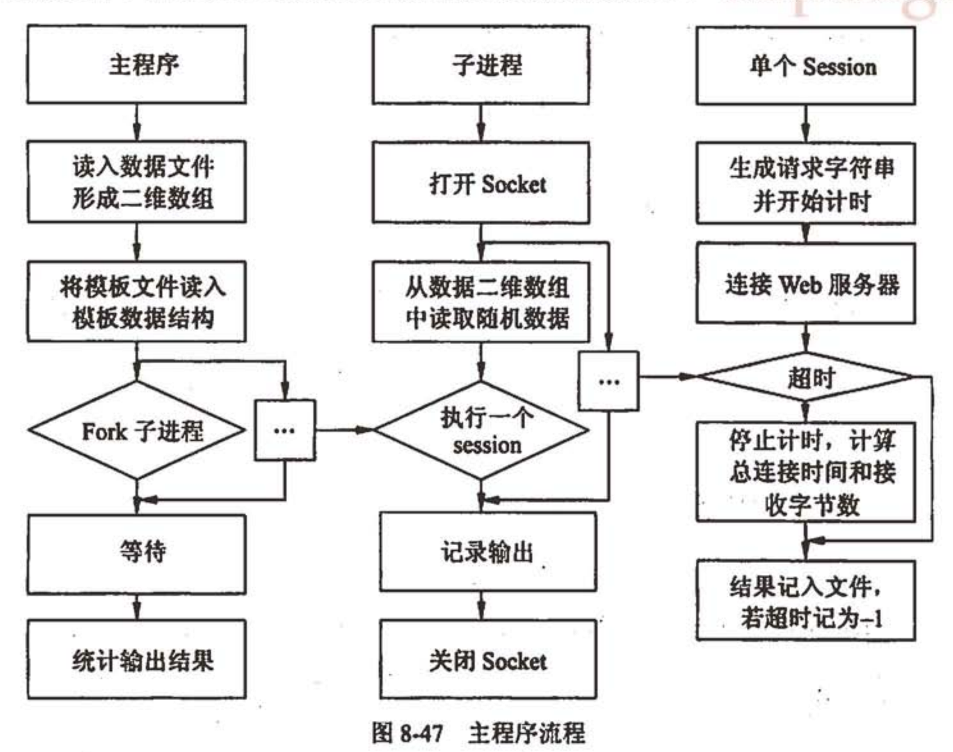


图 8-47 主程序流程

```
struct _tag_Template {
int count;              // 本模板中共包含的请求个数
char *method;           // 各请求发送的方法，0 = GET，1 = POST，
char **host;            // 各请求发送的目的主机名
int   *port;            // 各请求发送的目的主机端口号
char **path;            // 各请求要求的路径文件名
char **paramstr;        // 各请求包含的参数
}TEMPLATE;
```

测试程序运行结束后，生成的结果文件包含着 Web 服务器对每个请求的响应时间，还包含每个请求返回的字节数。结果文件由结果处理程序处理，计算出 Web 服务器对所有请求的平均响应时间。

（2）案例二：通用应用系统性能评测环境设计

大家知道 LoadRunner 通过运行脚本模拟的方法仍然与应用系统的实际处理逻辑存在一定的差异，因此其评测结果与应用系统实际使用过程中的性能指标之间仍然存在一定的差距。为了弥补这个缺陷，需要一种适用于特定应用的运行模拟和性能评测方法与

支撑环境，来对应用系统的实际性能评测提供有效支持。这里提出的通用应用系统性能评测环境，是通过并行执行分布于不同客户机上的多种类型的实际应用程序代码，模拟一个应用系统在多个客户端并发访问情况下的实际运行情况，同时对应用程序代码的执行状态和结果等信息进行记录；在模拟执行完成之后，对执行记录进行分析，给出被测系统的性能指标。应用系统性能评测环境主要由应用逻辑运行模拟代理（PexpAgent）、模拟控制中心（PexpCmdCenter）、数据搜集器（Data Collector）和统计分析工具等部分组成，其系统结构如图 8-48 所示。其中，应用逻辑运行模拟代理负责驱动执行被测系统的应用逻辑代码，并记录每一次执行的状态和性能相关信息。模拟控制中心负责控制整个性能评测过程中的所有运行模拟代理。数据搜集器负责在测量完成之后，接收来自运行模拟代理的性能测量数据。统计分析工具负责对测量获得的性能相关数据进行分析，从而得出被测系统的相关性能数据。在评测过程中，运行模拟代理与控制中心之间通过基于对象传输的专用网络通信接口（Performance Network）进行通信；由于应用系统的实际运行环境各异，评测系统被设计成能够运行在多种操作系统平台上，所有其他组件的下层是操作系统相关调用抽象层，对于不同的操作系统平台，需要实现相应的实际处理对象。

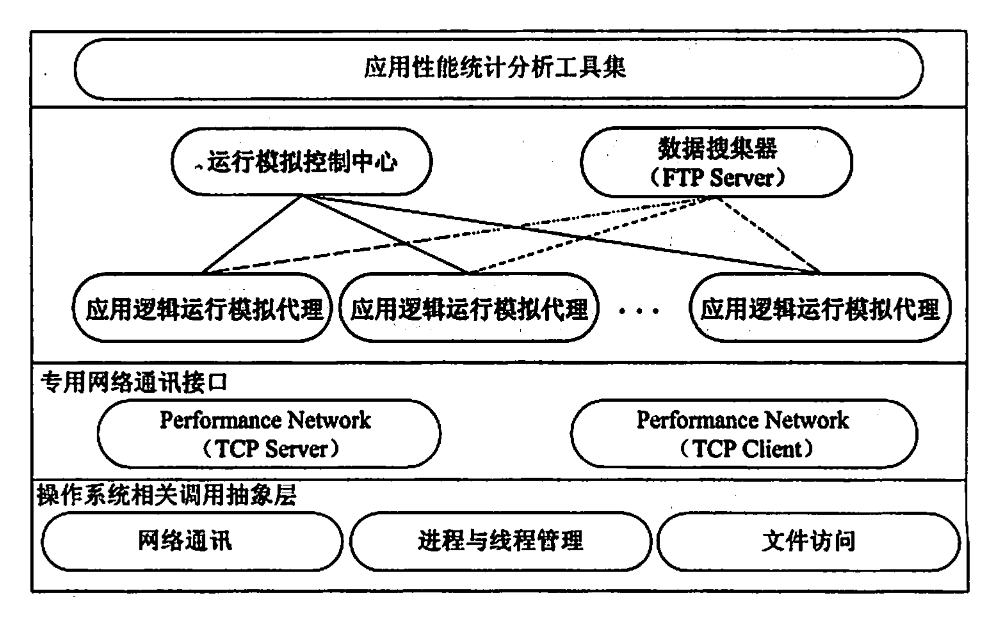


图 8-48 系统结构图

由于我们的目的是通过模拟真实的并发环境来获取系统的实际性能，在设计与实现上需要充分考虑如下问题，并加以解决：

- 应用系统实际处理代码的嵌入方式和响应时间的测定；
- 控制中心对运行模拟代理的有效控制；
- 多操作系统平台支持与评测系统本身的执行效率；
- 可扩展性。

6．准备过程中与用户交流

做为第三方测试机构，在负载压力测试准备的过程中，经常需要与用户交流，那么交流哪些内容呢？概括如下。

- 系统的运行模式；
- 系统运行的硬件环境和软件环境，包括客户端、中间件、Web 服务器、应用服务器、数据库服务器等；
- 系统网络架构；
- 系统主要功能模块以及主要功能需求；
- 系统主要性能需求，例如并发、疲劳、大数量等；
- 系统未来发展功能需求和性能需求；
- 系统网络容量规划等。

8.4.6 测试脚本录制、编写与调试

测试脚本指 Vuser 脚本，即虚拟用户回放所使用的脚本。脚本的产生可以采用录制、编写或者录制加编写混合模式，初始生成的脚本经过增强编辑之后，必需再经调试才可用。

Vuser 脚本的结构和内容因 Vuser 类型的不同而不同。例如，数据库 Vuser 脚本总是包含三部分，是在一段类似 C 语言并且包括对数据库服务器的 SQL 调用的代码中编写的。相反，GUI Vuser 脚本只有一个部分，并且是用 TSL（测试脚本语言）编写的。图 8-49 概述了开发 Vuser 脚本的过程。

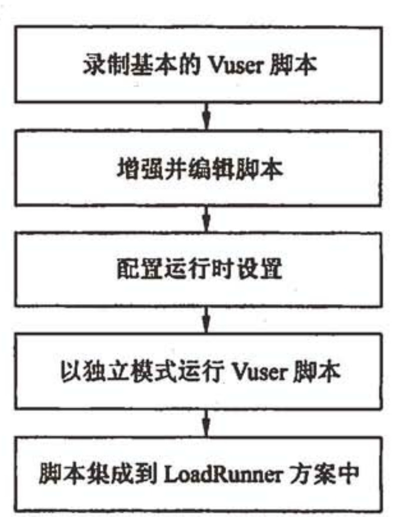


图 8-49 开发 Vuser 脚本的过程

首先，来了解录制脚本。在一般的测试过程中，录制脚本所占比例较大，测试工具提供了大量录制 Vuser 脚本的工具，并且可以通过将控制流结构和其他测试工具的 API 添加到脚本中来增强该基本脚本。然后，配置运行时设置。运行时设置包括迭代、日志和计时信息以及定义 Vuser 在执行 Vuser 脚本时的行为。要验证脚本是否能正确运行，请以单独模式运行该脚

本。如果脚本运行正确，则将其合并到方案中。

那么，录制哪些内容呢？主要录制用户在客户端应用程序中执行的典型业务流程。测试工具通过录制客户端和服务器之间的活动来创建脚本。例如，在数据库应用程序中，测试工具的脚本生成器（VuGen）会监控数据库的客户端，并跟踪发送到数据库服务器和从数据库服务器接收的所有请求。如图 8-50 所示为录制活动创建脚本。

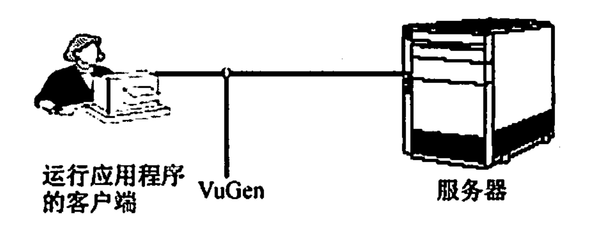


图 8-50 录制活动创建脚本

用 VuGen 创建的每个 Vuser 脚本都可以通过执行对服务器 API 的调用来直接与服务器通信，而不需要依赖客户端软件。这样，便可以使用 Vuser 来检查服务器性能（甚至在客户端软件的用户界面完全开发好之前）。如图 8-51 所示为 API 调用生成脚本。

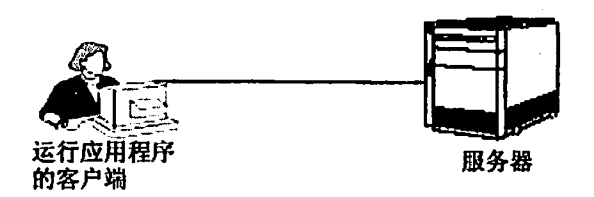


图 8-51 API 调用生成脚本

此外，当 Vuser 与服务器直接通信时，不需要在用户界面中耗费系统资源。这样就可以在一个工作站中同时运行大量 Vuser，进而可以使用很少的测试计算机来模拟非常大的服务器负载。

测试工具都留有手工编写脚本的入口，例如 C script、Java script、VB 以及汇编语言等，并且提供相应测试类型的 API，测试人员在此环境下可以编程生成脚本。

脚本的调试也是非常重要的工作，例如我们要调试 C/S 脚本，那么应该注意些什么呢？

对于 C/S 结构的脚本，在数据量大时，脚本非常庞大，如果全部看一遍，根本是不可能的。对于这种脚本的调试，应注意以下几个方面。

- 动态数据的处理。

我们经常会碰到某个表单的编号是记录在另一个表中的，程序通过查询这个表，并



加 1 来获取到这个编号。对于这种问题，可以分解为以下 3 步（以 ORACLE 数据库为例）。

① 获取数据，可使用 lrd_ora8_save_col 函数。

② 函数值加 1 处理，可使用 lr_param_increment 函数。

③ 替换处理，即把 Update 中的具体值替换为我们获取并处理好的参数就可以了。

- 参数化过程。

这一过程，我们所关注的，不过是 Insert 及 Update 语句。将这些语句中违反数据库约束的地方进行参数化就可以了。而且仅关注这些语句，基本上就可以搞清楚整个程序的处理流程。理消关系，作参数时直接 Replace All（录制脚本时注意使用的数据最好有特点，这样替换过程中就不会把不该替换的也替换了）就可以了。

8.4.7 场景制定

1. 创建 Vuser 组

方案由 Vuser 组构成，Vuser 模拟与应用程序进行交互的实际用户。运行方案时，Vuser 会在服务器上生成负载，测试工具会监视服务器和事务性能。Vuser 组用于将方案中的 Vuser 组织成可管理的组。可以创建包含具有共享或相似特征的 Vuser 的 Vuser 组。例如，可以为运行相同 Vuser 脚本的所有 Vuser 创建 Vuser 组。

2. 配置 Vuser 组中的 Vuser

可以为定义的 Vuser 组中的各个 Vuser 定义属性。对于每个 Vuser，可以分配不同的脚本和负载生成器计算机。

3. 配置 Vuser 运行时的设置

可以设置脚本的运行时设置，采用在控制中心自定义执行 Vuser 脚本的方式。

4. 配置负载生成器

在测试执行之前，需要配置方案的负载生成器和 Vuser 行为，即制定场景。虽然默认设置与大多数环境对应，但是 LoadRunner 允许修改这些设置以便自定义方案行为。这些设置适用于所有未来的方案运行并且通常只需设置一次。这一类设置适用于方案中所有的负载生成器。如果全局方案设置与单个负载生成器的设置不同，则负载生成器设置将替代它们。

可以指出哪些负载生成器将在方案中运行 Vuser。例如，如果某个负载生成器不适用于特定方案，可以暂时排除此负载生成器。如果要隔离特定计算机以测试其性能，则禁用负载生成器相当有用。

可以为各个负载生成器配置附加设置。可以配置的设置有：状态、运行时文件存储、UNIX 环境、运行时配额、Vuser 状态、Vuser 限制、连接日志（专家模式）、防火墙和 WAN 仿真。

5. 配置终端服务设置

可以使用终端服务管理器，来远程管理在终端服务器上的、负载测试方案中运行的多个负载管理器。此外，可以使用终端服务器克服只能在基于 Windows 的负载生成器上运行单个 GUI Vuser 的局限性。通过为每个 GUI Vuser 打开一个终端服务器会话，可以在同一应用程序上运行多个 GUI Vuser。

使用终端服务，可以集中管理连接到服务器的每个客户端的计算资源，并为每名用户提供他们自己的工作环境。使用终端服务器客户端，可以通过远程计算机在基于服务器的计算环境中操作。终端服务器通过网络传送应用程序，并通过终端仿真软件显示它们。每个用户会登录并只会看到他们各自的会话，服务器操作系统以透明的方式将该会话独立于其他任何客户端会话进行管理。检查如图 8-52 所示的测试工具组件协同工作可以了解测试工具组件在终端会话期间如何协同工作。图 8-52 为测试工具组件协同工作示意。

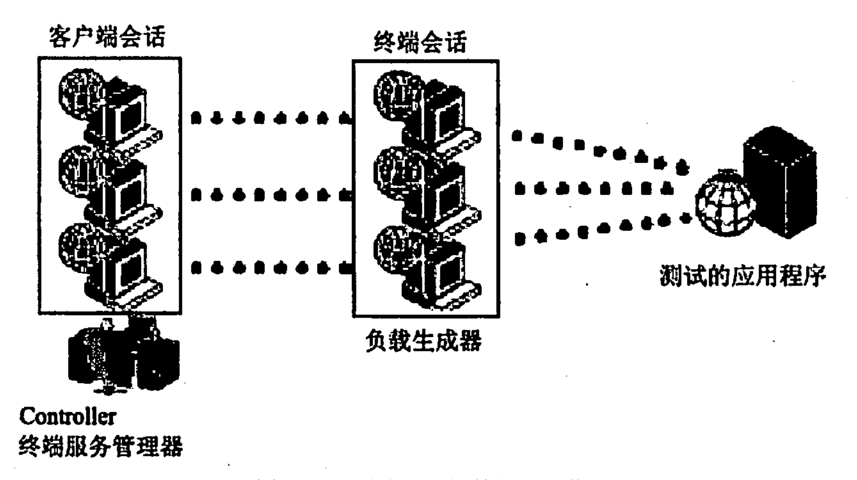


图 8-52　测试工具组件协同工作

终端服务器客户端可以同时运行多个终端会话。使用终端服务管理器，可以选择要在方案中使用的终端数量（如果有足够的终端会话在运行）以及每个终端可以运行的最大 Vuser 数。这样，终端服务管理器便可以在客户端会话间均匀地分配虚拟用户的数量。使用终端服务管理器可以做到以下几点。

- 在负载生成器计算机上设置终端服务器代理；
- 在控制中心计算机上启动终端客户端会话；
- 使用终端服务管理器在终端服务器上分配 Vuser。

6. 配置 WAN 仿真设置

可以使用 Shunra WAN 仿真器在负载测试方案中模拟各种网络基础结构的行为。使



用 WAN 仿真，可以在部署前模拟并测试广域网（WAN）对最终用户响应时间和性能的影响。

使用 WAN 仿真，可以在测试环境中准确地测试实际网络条件下 WAN 部署产品的点到点的性能。通过引入极为可能发生的 WAN 影响（如局域网中的滞后时间、包丢失、链路故障和动态路由等影响），可以描绘 WAN 云图的许多特征，并在单一网络环境中有效地控制仿真。可以在 WAN 仿真监视报告中观察仿真设置对网络性能的影响。

7．配置脚本

为 Vuser 或 Vuser 组选择了脚本后，可以编辑脚本或查看所选脚本的详细信息。

8.4.8 测试执行

1．运行场景

运行场景时，会为 Vuser 组分配负载生成器并执行它们的 Vuser 脚本。在场景执行期间，将要完成以下工作：

- 记录在 Vuser 脚本中定义的事务的持续时间；
- 执行包括在 Vuser 脚本中的集合；
- 收集 Vuser 生成的错误、警告和通知消息。

可以在无人干预的情况下运行整个场景，或者可以交互地选择要运行的 Vuser 组和 Vuser。场景开始运行时，Controller 会首先检查场景配置信息。接着，它将调用已选定与该场景一起运行的应用程序。然后，它会将每个 Vuser 脚本分配给其指定的负载生成器。Vuser 组就绪后，它们将开始执行其脚本。

在场景运行时，可以监视每个 Vuser，查看由 Vuser 生成的错误、警告和通知消息以及停止 Vuser 组和各个 Vuser。可以允许单个 Vuser 或组中的 Vuser 在停止前完成它们正在运行的迭代，在停止前完成它们正在运行的操作或者立即停止运行，还可以在场景运行时激活其他 Vuser。在下面情况下，场景将结束：所有 Vuser 已完成其脚本、持续时间用完或者终止场景。以下过程概述如何运行场景。

- 打开现有场景或新建一个场景；
- 配置并计划场景；
- 设置结果目录；
- 运行并监视场景。

2．在执行期间查看 Vuser

可以在场景执行期间查看 Vuser 的活动：

- 在 Controller 负载生成器计算机中，可以查看输出窗口，联机监视 Vuser 性能以及查看执行场景的 Vuser 的状态；

- 在远程计算机中，可以查看包含活动 Vuser 的有关信息的代理摘要。

3．监视场景

工具一般提供下列联机监视器：

- “运行时”监视器显示参与场景的 Vuser 的数目和状态，以及 Vuser 所生成的错误数量和类型。此外还提供用户定义的数据点图，其中显示 Vuser 脚本中的用户定义点的实时值。
- “事务”监视器显示场景执行期间的事务速率和响应时间。
- “Web 资源”监视器用于度量场景运行期间 Web 服务器上的统计信息。它提供关于场景运行期间的 Web 连接、吞吐量、HTTP 响应、服务器重试和下载页的数据。
- “系统资源”监视器测量场景运行期间使用的 Windows、UNIX、TUXEDO、SNMP 和 Antara FlameThrower 资源。要激活系统资源监视器，必须在运行场景之前设置监视器选项。
- “网络延迟”监视器显示关于系统上的网络延迟的信息。要激活网络延迟监视器，必须在运行场景之前设置要监视的网络路径。
- “防火墙”监视器用于度量场景运行期间防火墙服务器上的统计信息。要激活防火墙监视器，必须在运行场景之前设置要监视的资源列表。
- “Web 服务器资源”监视器用于度量场景运行期间 Apache、Microsoft IIS、iPlanet（SNMP）和 iPlanet/Netscape Web 服务器上的统计信息。要激活该监视器，必须在运行场景之前设置要监视的资源列表。
- “Web 应用程序服务器资源”监视器用于度量场景运行期间 Web 应用程序服务器上的统计信息。要激活该监视器，必须在运行场景之前设置要监视的资源列表。
- “数据库服务器资源”监视器用于度量与 SQL Server、Oracle、Sybase 和 DB2 数据库有关的统计信息。要激活该监视器，必须在运行场景之前设置要监视的度量列表。
- “流媒体”监视器用于度量 Windows Media 服务器、RealPlayer 音频/视频服务器及 RealPlayer 客户端上的统计信息。要激活该监视器，必须在运行场景之前设置要监视的资源列表。
- “ERP/CRM 服务器资源”监视器用于度量场景运行期间 SAP R/3 系统服务器、SAP Portal、Siebel Web 服务器和 Siebel Server Manager 服务器的统计信息。要激活该监视器，必须在运行场景之前设置要监视的资源列表。
- “Java 性能”监视器用于度量 Java 2 Platform, Enterprise Edition（J2EE）对象及使用 J2EE 和 EJB 服务器计算机的 Enterprise Java Bean（EJB）对象的统计信息。

要激活该监视器，必须在运行场景之前设置要监视的资源列表。

- “应用程序部署解决场景”监视器用于度量场景运行期间 Citrix MetaFrame XP 和 1.8 服务器的统计信息。要激活该监视器，必须在运行场景之前设置监视器选项。
- “中间件性能”监视器用于度量场景运行期间 TUXEDO 和 IBM WebSphere MQ 服务器上的统计信息。要激活该监视器，必须在运行场景之前设置要监视的资源列表。
- 所有的监视器所收集的数据都可以生成该监视器的图。

有些工具也提供远程性能监控。

在负载测试运行过程中，远程性能监视器可以查看特定的图，这些图显示 Vuser 在服务器上生成的负载的信息。用户在连接到 Web 服务器的 Web 浏览器上查看负载测试数据。如图 8-53 所示为利用远程性能监视器查看负载测试数据。

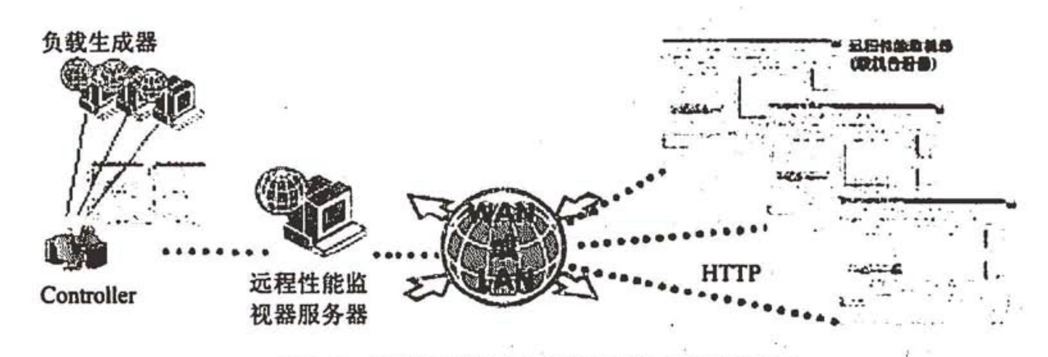


图 8-53 利用远程性能监视器查看负载测试数据

远程性能监视器服务器包含一个用 ASP 页实现的网站，以及一个包含负载测试图的文件服务器。它与 Controller 联机组件进行交互，并按相应的许可证处理同时查看负载测试的用户数。

8.4.9 获取测试结果

在场景执行期间，Vuser 会在执行事务的同时生成结果数据。要在测试执行期间监视场景性能，可以使用联机监视工具。要查看测试执行之后的结果摘要，可以使用下列一个或多个工具。

- “Vuser 日志文件”包含对每个 Vuser 运行的场景的完整跟踪。这些文件位于方案结果目录中（在以独立模式运行 Vuser 脚本时，这些文件放在 Vuser 脚本目录中）。

- “Controller 输出”窗口显示有关场景运行的信息。如果场景运行失败，可以在该窗口中查找调试信息。
- “Analysis 图”有助于确定系统性能并提供有关事务和 Vuser 的信息。通过合并几个场景的结果或者将几个图合并成一个图，可以对多个图进行比较。
- “图数据”视图和“原始数据”视图以电子表格格式显示用于生成图的实际数据。可以将这些数据复制到外部电子表格应用程序，以进行进一步处理。
- “报告”实用程序允许查看每个图的摘要 HTML 报告或各种性能和活动报告。可以将报告创建成 Microsoft Word 文档，它会自动以图形或表格形式总结和显示测试的重要数据。

工具的结果分析功能是有限的，要定位问题测试，工程师的经验和智慧应该起到很大的作用。如何定位问题，“测试实例”部分有案例介绍。

8.4.10　结果评估与测试报告

1．交易处理性能评估

交易处理性能评估指标主要包括：

- 并发用户数。

并发用户数是负载压力测试的主要指标，体现了系统能够承受的并发性能。

测试重点得到两类并发用户数指标，一类是系统最佳性能的并发用户数，另一类是系统能够承受的最大并发用户数，这两类指标在某种情况下有可能重叠。

- 交易响应时间。

该指标描述交易执行的快慢程度，这是用户最直接感受到的系统性能，也是故障定位迫切需要解决的问题。

- 交易通过率。

指每秒钟能够成功执行的交易数，描述系统能够提供的“产量”，用户可以以此来评估系统的性能价格比。

- 吞吐量。

指每秒通过的字节数，以及通过的总字节数。此指标在很大程度上影响系统交易的响应时间，形成响应时间的“拐点”。

- 点击率。

描述系统响应请求的快慢。

2．资源占用性能评估

资源占用主要涉及服务器操作系统资源占用、数据库资源占用、中间件资源占用等内容，下面分别论述。



（1）服务器操作系统资源占用

通过《负载压力测试指标》章节的讨论，可以将服务器操作系统资源占用监控指标概括为以下几个方面：

① CPU。

② 磁盘管理。

③ 内存。

④ 交换区 SWAP。

⑤ 进程。

⑥ 安全控制。

⑦ 文件系统。

下面举例对某些指标进行分析。

- Memory：内存使用情况可能是系统性能中最重要的因素。如果系统“页交换”频繁，说明内存不足。“页交换”是使用称为“页面”的单位，将固定大小的代码和数据块从 RAM 移动到磁盘的过程，其目的是为了释放内存空间。尽管某些页交换使 Windows 2000 能够使用比实际更多的内存，也是可以接受的，但频繁的页交换将降低系统性能。减少页交换将显著提高系统响应速度。要监视内存不足的状况，请从以下的对象计数器开始。

① Available Mbytes：可用物理内存数。如果 Available Mbytes 的值很小（4MB 或更小），则说明计算机上总的内存可能不足，或某程序没有释放内存。

② page/sec：表明由于硬件页面错误而从磁盘取出的页面数，或由于页面错误而写入磁盘以释放内存空间的页面数。一般如果 pages/sec 持续高于几百，那么应该进一步研究页交换活动。有可能需要增加内存，以减少换页的需求（你可以把这个数字乘以 4k 就得到由此引起的硬盘数据流量）。pages/sec 的值很大，不一定表明内存有问题，而可能是运行使用内存映射文件的程序所致。

③ page read/sec：页的硬故障，page/sec 的子集，为了解析对内存的引用，必须读取页文件的次数。阈值为>5，越低越好。大数值表示磁盘读而不是缓存读。

- 由于过多的页交换要使用大量的硬盘空间，因此有可能导致页交换内存不足与页交换的磁盘瓶颈混淆。因此，在研究内存不足不太明显的页交换的原因时，必须跟踪如下的磁盘使用情况计数器和内存计数器：
- Physical Disk\ % Disk Time。
- Physical Disk\ Avg.Disk Queue Length。例如，包括 Page Reads/sec 和% Disk Time 及 Avg.Disk Queue Length。如果页面读取操作速率很低，同时%Disk Time 和 Avg.Disk Queue Length 的值很高，则可能有磁盘瓶颈。而如果队列长度增

加的同时页面读取速率并未降低，则内存不足。要确定过多的页交换对磁盘活动的影响，请将 Physical Disk\ Avg.Disk sec/Transfer 和 Memory\ pages/sec 计数器的值增大数倍。如果这些计数器的计数结果超过了 0.1，那么页交换将花费 10%以上的磁盘访问时间。如果长时间发生这种情况，那么可能需要更多的内存。

- Page Faults/sec：每秒钟软性页面失效的数目（包括有些可以直接在内存中满足而有些需要从硬盘读取），而 page/sec 只表明数据不能在指定内存中立即使用。
- Cache Bytes：文件系统缓存（File System Cache），默认情况下为 50%的可用物理内存。如果怀疑有内存泄露，请监视 Memory\ Available Bytes 和 Memory\ Committed Bytes，以观察内存行为，并监视可能泄露内存进程 Process\Private Bytes、Process\Working Set 和 Process\Handle Count。如果怀疑是内核模式进程导致了泄露，则还应该监视 Memory\Pool Nonpaged Bytes、Memory\ Pool Nonpaged Allocs 和 Process(process_name)\ Pool Nonpaged Bytes。
- Pages per second：每秒钟检索的页数。该数字应少于每秒 1 页。
- Page Faults/sec：将进程产生的页故障与系统产生的相比较，以判断这个进程对系统页故障产生的影响。
- Work set：处理线程最近使用的内存页，反映了每一个进程使用的内存页的数量。如果服务器有足够的空闲内存，页就会被留在内存中，当自由内存少于一个特定的阈值时，页就会被清除出内存。
- Inetinfo：Private Bytes。此进程所分配的无法与其他进程共享的当前字节数量。如果系统性能随着时间而降低，则此计数器可以是内存泄漏的最佳指示器。
- Processor：监视“处理器”和“系统”对象计数器可以提供关于处理器使用的有价值的信息，帮助决定是否存在瓶颈。
- %Processor Time：被处理器消耗的处理器时间数量。如果该值持续超过 95%，表明瓶颈是 CPU。可以考虑增加一个处理器或换一个更快的处理器。
- %User Time：表示耗费 CPU 的数据库操作，如排序，执行 aggregate functions 等。如果该值很高，可考虑增加索引，尽量使用简单的表联接、水平分割大表格等方法来降低该值。
- %Privileged Time：（CPU 内核时间）是在特权模式下处理线程执行代码所花时间的百分比。如果该参数值和“Physical Disk”参数值一直很高，表明 I/O 有问题。可考虑更换更快的硬盘系统。另外设置“Tempdb in RAM”，减低“max async IO”，“max lazy writer I/O”等措施都会降低该值。此外，跟踪计算机的服务器工作队列当前长度的 Server Work Queues\ Queue Length 计数器会显示出处理器



瓶颈。队列长度持续大于 4，则表示可能出现处理器拥塞。此计数器是特定时间的值，而不是一段时间的平均值。

- % DPC Time：越低越好。在多处理器系统中，如果这个值大于 50%并且"Processor：% Processor Time"非常高，加入一个网卡可能会提高性能，提供的网络已经不饱和。
- Context Switches/sec：如果决定要增加线程字节池的大小，应该同时监视实例化 inetinfo 和 dllhost 进程这两个计数器。增加线程数可能会增加上下文切换次数，这样性能不会上升，反而下降。如果多个实例的上下文切换值非常高，就应该减小线程字节池的大小。
- %Disk Time：指所选磁盘驱动器忙于为读或写入请求提供服务所用的时间的百分比。如果只有%Disk Time 比较大，而 CPU 和内存都比较适中，硬盘可能会是瓶颈。
- Avg.Disk Queue Length：指读取和写入请求（为所选磁盘在实例间隔中列队）的平均数。该值应不超过磁盘数的 1.5～2 倍。要提高性能，可增加磁盘。注意，一个 Raid Disk 实际有多个磁盘。
- Average Disk Read/Write Queue Length：指读取（写入）请求（列队）的平均数。
- Disk Reads（Writes）/s：物理磁盘上每秒钟磁盘读、写的次数。两者相加，应小于磁盘设备最大允许读取次数。
- Average Disksec/Read：指以秒计算的在此盘上读取数据的所需平均时间。
- Average Disk sec/Transfer：指以秒计算的在此盘上写入数据的所需平均时间。
- Bytes Total/sec：为发送和接收字节的速率，包括帧字符在内。判断网络连接速度是否是瓶颈，可以用该计数器的值和目前网络的带宽进行比较。

（2）数据库资源占用监控指标包括

通过《负载压力测试指标》章节的讨论，可以将数据库资源占用监控指标概括为：

- 读写页面的使用情况。
- 超出共享内存缓冲区的操作数。
- 上一轮询期间作业等待缓冲区的时间。
- 共享内存中物理日志和逻辑日志的缓冲区的使用率。
- 磁盘的数据块使用情况以及被频繁读写的热点区域。
- 用户事务或者表空间事务。
- 数据库锁资源。
- 关键业务的数据表的表空间增长。
- SQL 执行情况。

下面以 SQL Server 数据库性能计数器为例来分析。

- Access Methods：用于监视数据库逻辑页访问方法。
- Full Scans/sec：每秒钟不受限的完全扫描数。可以是基本表扫描或全索引扫描。如果这个计数器显示的值比 1 或 2 高，应该分析查询以确定是否确实需要全表扫描，以及 SQL 查询是否可以被优化。
- Page splits/sec：由于数据更新操作引起的每秒页分割的数量。
- Buffer Manager：监视 SQL Server 如何使用内存存储数据页、内部数据结构和过程高速缓存。计数器在 SQL Server 从磁盘读取数据库页和将数据库页写入磁盘时监视物理 I/O。监视 SQL Server 所使用的内存和计数器，有助于确定是否由于缺少可用物理内存存储高速缓存中经常访问的数据，而导致瓶颈存在。如果是这样，SQL Server 必须从磁盘检索数据，以及考虑是否可通过添加更多内存，或使更多内存可用于数据高速缓存或 SQL Server 内部结构来提高查询性能。
- Disk I/O：SQL Server 从磁盘读取数据的频率。与其他操作相比，例如内存访问，物理 I/O 会耗费大量时间。尽可能减少物理 I/O 可以提高查询性能。
- Page Reads/sec：每秒钟发出的物理数据库页读取数。这一统计信息显示的是在所有数据库间的物理页读取总数。由于物理 I/O 的开销大，可以通过使用更大的数据高速缓存、智能索引、更高效的查询或者改变数据库设计等方法，使开销减到最小。
- Page Writes/sec：每秒执行的物理数据库写的页数。
- Buffer Cache Hit Ratio：在“缓冲池”（Buffer Cache/Buffer Pool）中没有被读过的页占整个缓冲池中所有页的比率。可在高速缓存中找到，而不需要从磁盘中读取的页的百分比。这一比率是高速缓存命中总数除以自 SQL Server 实例启动后对高速缓存的查找总数。经过很长时间后，这一比率的变化很小。由于从高速缓存中读数据比从磁盘中读数据的开销要小得多，一般希望这一数值高一些。通常，可以通过增加 SQL Server 可用的内存数量来提高高速缓存命中率。计数器值依应用程序而定，但比率最好为 90% 或更高。增加内存直到这一数值持续高于 90%，表示 90%以上的数据请求可以从数据缓冲区中获得所需数据。
- Lazy Writes/sec：惰性写进程每秒写的缓冲区的数量。其值最好为 0。
- Cache Manager：对象提供计数器，用于监视 SQL Server 如何使用内存存储对象，如存储过程、特殊和准备好的 Transact-SQL 语句以及触发器。
- Cache Hit Ratio：Cache 可以包括 Log Cache，Buffer Cache 以及 Procedure Cache，



是一个总体的比率，是高速缓存命中次数和查找次数的比率之和。其对查看 SQL Server 高速缓存对于系统性能提升如何有效，是一个非常好的计数器。如果这个值持续低于 80%，就需要增加更多的内存。

- Latches：用于监视称为"闩锁"的内部 SQL Server 资源锁。监视闩锁以明确用户活动和资源使用情况，有助于查明性能瓶颈。
- Average Latch Wait Time (ms)：一个 SQL Server 线程必须等待一个闩的平均时间，以毫秒为单位。如果这个值很高，系统可能正经历严重的竞争问题。
- Latch Waits/sec：在闩上每秒的等待数量。如果这个值很高，表明系统正经历严重的竞争问题。
- Locks：提供有关个别资源类型上的 SQL Server 锁的信息。锁加在 SQL Server 资源上(如在一个事务中进行的行读取或修改)，以防止多个事务并发使用资源。例如，如果一个排它锁被一个事务加在某一表的某一行上，在这个锁被释放前，其他事务都不可以修改这一行。应尽可能少使用锁，可提高并发性，从而改善性能。可以同时监视 Locks 对象的多个实例，每个实例代表一个资源类型上的一个锁。
- Number of Deadlocks/sec：导致死锁的锁请求的数量。
- Average Wait Time(ms)：线程等待某种类型的锁的平均等待时间。
- Lock Requests/sec：每秒钟某种类型的锁请求的数量。
- Memory manager：用于监视总体的服务器内存使用情况，以估计用户活动和资源使用，有助于查明性能瓶颈。监视 SQL Server 实例所使用的内存，有助于确定是否由于缺少可用物理内存存储高速缓存中经常访问的数据而导致瓶颈存在。如果是这样，SQL Server 必须从磁盘检索数据，以及考虑是否可以通过添加更多内存，或使更多内存可用于数据高速缓存或 SQL Server 内部结构，来提高查询性能。
- Lock blocks：服务器上锁定块的数量，锁是加在页、行或者表这样的资源上。通常不希望看到此值增长。
- Total Server Memory：SQL Server 服务器当前正在使用的动态内存总量。

（3）中间件资源占用监控

中间件主要包括：

- Web 中间件；
- 应用中间件；
- 交易中间件；
- 其他中间件。

下面以 IIS 为例来分析。

- % File Cache Hits：是全部缓存请求中缓存命中次数所占的比例，反映了 IIS 的文件缓存设置的工作情况。对于一个由大部分静态网页组成的网站，该值应该保持在 80%左右。File Cache Hits 是文件缓存命中的具体值，而 File CacheFlushes 是自服务器启动之后文件缓存刷新次数，如果刷新得太慢，会浪费内存；如果刷新得太快，缓存中的对象会太频繁地丢弃生成，起不到缓存的作用。通过比较 File Cache Hits 和 File Cache Flushes 可得出缓存命中率与缓存清空率的比率。通过观察这两个值，可以得到一个适当的刷新值（参考 IIS 的 ObjectTTL、MemCacheSize 和 MaxCacheFileSize 设置）。
- Web Service 部分：

① Bytes Total/sec：显示 Web 服务器发送和接收的总字节数。低数值表明该 IIS 正在以较低的速度进行数据传输。

② Connection Refused：数值越低越好。高数值表明网络适配器或处理器存在瓶颈。

③ Not Found Errors：显示由于被请求文件无法找到而导致的服务器无法回应的请求数（HTTP 状态代码 404）。

3．故障分析

这里主要讨论故障分析内容以及优化调整设置内容，同时还与读者分享故障分析的经验与实例。

（1）故障分析重点内容

故障分析的重点内容包括以下几个方面：

① CPU 问题。

② 内存和高速缓存。

③ 磁盘（I/O）资源问题。

④ 配置参数。

⑤ 应用系统网络设置。

⑥ 数据库服务器故障定位。

（2）经验探讨

- 经验举例 1。

交易的响应时间如果很长，远远超过系统性能的需求，表示耗费 CPU 的数据库操作。例如排序，执行 aggregate functions（例如 sum、min、max、count）等较多，可考虑是否有索引以及索引建立得是否合理。尽量使用简单的表链接、水平分割大表格等方法来降低该值。

- 经验举例 2。



测试工具可以模拟不同的虚拟用户来单独访问 Web 服务器、应用服务器和数据库服务器，这样，就可以在 Web 端测出的响应时间减去以上各个分段测出的时间，就可以知道瓶颈在哪里并着手调优。

- 经验举例 3。

UNIX 资源监控（NT 操作系统同理）中指标内存页交换速率（Paging rate），如果该值偶尔走高，表明当时有线程竞争内存。如果持续很高，则内存可能是瓶颈，也可能是内存访问命中率低。“Swap in rate”和“Swap out rate”也有类似的解释。

- 经验举例 4。

UNIX 资源监控（NT 操作系统同理）中指标 CPU 占用率（CPU utilization），如果该值持续超过 95%，表明瓶颈是 CPU。可以考虑增加一个处理器或换一个更快的处理器。合理使用的范围在 60%～70%。

- 经验举例 5。

Tuxedo 资源监控中指标队列中的字节数（Bytes on queue），队列长度应不超过磁盘数的 1.5~2 倍。要提高性能，可增加磁盘。注意：一个 Raid Disk 实际有多个磁盘。

- 经验举例 6。

SQL Server 资源监控中指标缓存点击率（Cache Hit Ratio），该值越高越好。如果持续低于 80%，应考虑增加内存。注意该参数值是从 SQL Server 启动后，就一直累加记数，所以运行经过一段时间后，该值将不能反映系统当前值。

（3）优化调整设置

针对上述故障分析的重点内容，需要做相应的优化调整，建议如下。

- CPU 问题。

① 考虑使用更高级的 CPU 代替目前的 CPU；

② 对于多 CPU，考虑 CPU 之间的负载分配；

③ 考虑在其他体系上设计系统，例如增加前置机、设置并行服务器等。

- 内存和高速缓存。

① 内存的优化包括操作系统、数据库、应用程序的内存优化；

② 过多的分页与交换可能降低系统的性能；

③ 内存分配也是影响系统性能的主要原因；

④ 保证保留列表具有较大的邻接内存块；

⑤ 调整数据块缓冲区大小（用数据块的个数表示）是一个重要内容；

⑥ 将最频繁使用的数据保存在存储区中。

- 磁盘（I/O）资源问题。

① 磁盘读写进度对数据库系统是至关重要的，数据库对象在物理设备上的合理分

布能改善性能。

② 磁盘镜像会减慢磁盘写的速度。

③ 通过把日志和数据库对象分布在独立的设备上，可以提高系统的性能。

④ 把不同的数据库放在不同的硬盘上，可以提高读写速度。建议把数据库、回滚段、日志放在不同的设备上。

⑤ 把表放在一块硬盘上，把非簇的索引放在另一块硬盘上，保证物理读写更快。

- 调整配置参数。

① 包括操作系统和数据库的参数配置。

② 并行操作资源限制的参数（并发用户的数目、会话数）。

③ 影响资源开销的参数。

④ 与 I/O 有关的参数。

- 优化应用系统网络设置。

① 可以通过数组接口来减少网络呼叫。不是一次提取一行，而是在单个往来往返中提取 10 行，这样做效率较高。

② 调整会话数据单元的缓冲区大小。

③ 共享服务进程比专用服务进程提供更好的性能。

（4）负载压力典型问题分析

负载压力测试需要识别的故障问题主要包括：

- 非正确执行的处理。
- 速度瓶颈与延迟。
- 不能达到满意服务水平。
- 接口页面不能正确地装载或者根本不能装载。

当在合理的加载下出现这些类型的问题时，则表示可能有基础性的设计问题，比如说：算法问题，低效的数据库应用程序交互作用等，这些都不是通过简单升级硬件以及调整系统配置就可以解决的问题，此时软件的故障定位和调优将占有更重要的地位。

（5）Web 网站故障分析举例

目前 Web 开发者开始提供可定制的 Web 网站，例如，像搜索数据之类的任务，现在可以由服务器执行，而无需客户干预。然而，这些变革也导致了一个结果，这就是许多网站都在使用大量的未经优化的数据库调用，从而使得应用性能大打折扣。

我们可以使用以下几种方法来解决这些问题：

- 优化 ASP 代码。
- 优化数据库调用。



- 使用存储过程。
- 调整服务器性能。

优秀的网站设计都会关注这些问题。然而，与静态页面的速度相比，任何数据库调用都会显著地影响 Web 网站的响应速度，这主要是因为在发送页面之前必须单独地为每个访问网站的用户进行数据库调用。

这里提出的性能优化方案正是基于以下事实：访问静态 HTML 页面要比访问那些内容依赖于数据库调用的页面要快。它的基本思想是：在用户访问页面之前，预先从数据库提取信息，写入存储在服务器上的静态 HTML 页面。为了保证这些静态页面能够及时地反映不断变化的数据库数据，必须有一个调度程序管理静态页面的生成。

当然，这种方案并不能够适应所有的情形。例如，如果是从持续变化的大容量数据库提取少量信息，这种方案是不合适的。

每当该页面被调用时，脚本就会提取最后的更新时间并将它与当前时间比较。如果两个时间之间的差值大于预定的数值，更新脚本就会运行，否则，该 ASP 页面把余下的 HTML 代码发送给浏览器。

如果每次访问 ASP 页面的时候都要提供最新的信息，或者输出与用户输入密切相关，这种方法并不实用，但这种方法可以适应以固定的时间间隔更新信息的场合。

如果数据库内容由客户通过适当的 ASP 页面更新，要确保静态页面也能够自动反映数据的变化，我们可以在 ASP 页面中调用 Update 脚本。这样，每当数据库内容改变时，服务器上也有了最新的静态 HTML 页面。

另一种处理频繁变动数据的办法是借助 Microsoft SQL Server 7.0 或以上版本的 Web 助手向导（Web Assistant Wizard），这个向导能够利用 Transact-SQL、存储过程等从 SQL Server 数据生成标准的 HTML 文件。

Web 助手向导能够用来定期地生成 HTML 页面。正如前面概要介绍的方案，Web 助手可以通过触发子更新 HTML 页面，比如在指定的时间执行更新或者在数据库数据变化时执行更新。

SQL Server 使用名为 sp_makewebtask 的存储过程创建 HTML 页面，它的参数是目标 HTML 文件的名字和待执行存储过程的名字，查询的输出发送到 HTML 页面。另外，也可以选择使用可供结果数据插入的模板文件。

万一用户访问页面的时候正好在执行更新，我们可以利用锁或者其他类似的机制把页面延迟几秒钟。

我们对纯 HTML 加调度 ASP 代码和普通的 ASP 文件进行了性能测试。普通的 ASP 文件要查找 5 个不同的表为页面提取数据。为了和这两个文件相比较，对一个只访问单个表的 ASP 页面和一个纯 HTML 文件也进行了测试。测试结果如表 8-25 所示。

表 8-25　加调度 ASP 代码和普通 ASP 代码测试对比

文 件 名 字	命 中 数	平均 TTFB（ms）	平均 TTLB（ms）
纯 HTML 文件	8	47	474
只访问单个表的 ASP 页面	8	68.88	789.38
普通的 ASP 文件	9	125.89	3759.56
纯 HTML 加调度 ASP 代码	9	149.89	1739.89

其中 TTFB 是指“Total Time to First Byte”，TTLB 是指“Total Time to Last Byte”。

测试结果显示，访问单个表的 ASP 页面的处理时间是 720.5ms，而纯 HTML 文件则为 427ms。普通的 ASP 文件和纯 HTML 加调度 ASP 代码的输出时间相同，但它们的处理时间分别为 3633.67ms 和 1590ms。也就是说，在这个测试环境下我们可以把处理速度提高 43%。

如果我们要让页面每隔一定的访问次数进行更新，比如 100 次，那么这第 100 个用户就必须等待新的 HTML 页面生成。不过，这个代价或许不算太高，其他 99 个用户获得了好处。

静态页面方法并不能够适合所有类型的页面。例如，某些页面在进行任何处理之前必须要有用户输入。但是，这种方法可以成功地应用到那些不依赖用户输入却进行大量数据库调用的页面，而且这种情况下它将发挥出更大的效率。

在大多数情况下，动态页面的生成将在相当大的程度上提高网站的性能，而且无须在功能上有所折衷。虽然有许多大的网站采用了这个策略来改善性能，但也有许多网站完全由于进行大量没有必要的数据库调用，而表现出很差的性能。

4．数据库服务器性能问题及原因分析

数据库服务器性能问题主要表现在某些类型操作的响应时间过长、同一类型事务的并发处理能力差和锁冲突频繁发生等方面。应该说，这些问题是数据库服务器性能不佳的典型表现。由于造成上述情况的原因众多，需要分情况加以分析。

（1）单一类型事务响应时间过长

响应时间（Response Time，RT）是系统完成事务执行准备后所采集的时间戳和系统完成待执行事务后所采集的时间戳之间的时间间隔，是衡量特定类型应用事务性能的重要指标，标志了用户执行一项操作大致需要多长时间。响应时间过长意味着用户完成执行一项命令需要等待相当长的时间。实践表明，通常情况下用户能够接受的响应时间最大为 200 ms。不仅如此，响应时间过长也是造成系统锁冲突严重的重要原因之一。

造成响应时间过长的原因非常复杂，通常可以从以下几个方面考虑。

① 数据库服务器负载过重。

② 糟糕的数据库设计。

③ 事务粒度过大。

④ 批任务对普通用户性能的影响。

- 数据库服务器负载过重。

数据库服务器负载过重不可避免地会造成响应时间过长。这标志着当前服务器系统的硬件条件不能满足实际用户对性能的需要。服务器负载过重主要表现在 CPU 使用率高、内存占用率大、I/O 与页面交换频繁发生等方面。由于服务器系统本身一般都提供性能监控程序，定位服务器性能问题相对比较容易。解决这类问题的方法一般是升级服务器硬件，提高数据库服务器本身的处理能力。但是，通过升级服务器硬件获得的性能提升是很有限的，并且对于某些诸如算法复杂度过高等问题根本无法解决。

- 糟糕的数据库设计。

糟糕的数据库设计是导致单一事务响应时间过长的最重要原因。通常，数据库设计在系统开发初期进行，此时，数据库设计人员往往对数据的实际规模和特性没有足够的了解。在数据库设计方面，对响应时间影响较大的因素有数据库表的规模、索引的使用、数据的分布、查询优化等。其中索引的使用极大影响事务执行的响应时间。实际上，很多情况下应用程序在访问大规模数据库表时的确没有使用索引。造成这种情况的原因通常是开发人员的疏忽，也有用户的需求变化过多，某些数据库字段不适合建立索引的情况。在这里需要特别指出的是，对组合索引进行查询时，查询条件中字段的顺序与数据库设计的索引字段顺序要一致。如果顺序颠倒，组合索引根本不能被数据库使用。例如，表 A（field1，field2，field3），有组合索引 index（field1，field2），诸如

SELECT * FROM A WHERE（(field2 = condition2）AND（field1 = condition1））和 SELECT * FROM A WHERE（field2 = condition2）的查询无法使用 index 索引。此外，并非增加索引就一定能够提高单一事务执行的响应时间。过多的索引使用将极大地增加插入操作（INSERT）的花费，从而使响应时间变长。

数据库表规模过大，是指单一数据库表的记录数在百万行以上，对这类表直接进行检索而不采取必要的优化手段，必然造成单一查询响应时间过长。如果数据库表的规模一再增大，使用索引也不能很有效地解决响应时间过长的问题。对此类问题，可能的解决办法是对数据进行分布，使查询能够并行执行或缩小查询的范围。目前，主流关系数据库管理系统都提供表分区（分段）存储，以使得软件开发人员比较容易地实现数据分布。

查询优化对响应时间的影响也不容忽视，尽管优化活动由数据库管理系统完成。实际上一条结构化查询语句(SQL)的写法有很多种，不同的写法可能有不同的响应时间，这一差别可能非常大。如果开发人员在软件编写过程中恰好使用了执行效率低的 SQL 语

句，其响应时间自然就会变长。目前，某些独立软件开发商已经注意到这种情况，并开发了相应的软件帮助应用软件开发人员找到执行最快的 SQL 语句的写法。但是，由于数据库本身是动态变化的，执行最快的 SQL 语句也可能变化，所以这种方法也是有局限性的。

- 事务粒度过大。

事务粒度过大指单一数据库事务执行过程中，需要以某种并发控制机制访问多个数据库资源。通常采用的并发控制机制是互斥锁或者共享锁。这种大粒度事务的执行由于要访问多个数据库资源（如数据库表），本身就需要消耗相当长的时间。此外，由于通常事务在执行的时候会对数据库资源进行加锁，这类事务也对其他访问该资源的用户造成影响。由于这类事务通常使用的锁数量都在两个以上，如果不合理地进行控制，极容易造成死锁。因此，在应用软件设计过程中，应该尽量消除大粒度事务。

- 批任务对普通用户性能的影响。

批任务是指一次操作将对数据库中大量数据进行互斥访问的数据库事务。这种类型的事务通常将更新同一个数据库表中的数千项乃至更多的数据。由于这类任务把所有操作放置在同一个数据库事务中，所访问的资源在其执行过程中始终被锁定，必然会对其他普通事务造成访问影响。此外，由于这类任务本身将对数据库服务器造成巨大的负担，使得服务器负载加重，从而影响独立事务的响应时间。通常情况下，批任务推荐在系统具有较长空闲时完成（如晚上），这样可以保证不对独立事务造成影响。如果由于业务的要求，批任务必须与独立事务混合运行，则必须对其加以改造，以减轻对其他事务的影响。

（2）并发处理能力差

并发处理能力差是指应用系统在执行同一类型事务的多个实例时，不能获得与执行实例数量相当的吞吐量，而是大大低于理论值。一般来说，这类问题都是由于互斥访问造成的，即并发执行中的某个实例以互斥方式对资源进行访问，造成了其他同类型用户必需等待该实例释放锁定资源后才能执行。应该指出，由于某些资源必须以互斥的方式进行访问，某些类型的事务在同一时间是只能有一个进行执行的。对于并发处理能力差的问题，可能的解决方法有，降低同一类型事务中锁的粒度、优化应用逻辑以缩短单一类型事务响应时间等。

（3）锁冲突严重

锁冲突是每个以关系数据库为核心的信息系统必须解决的问题。这里的锁冲突是指同一类型或不同类型事务在并发执行的情况下，由于资源互斥而相互影响，造成一个或多个事务无法正常执行的情况，包括资源锁定造成的数据库事务超时和死锁两个方面。

- 资源锁定造成的数据库事务超时。



资源锁定导致的数据库事务超时，其原因是多方面的，其中，批任务影响其他类型独立事务的情况占有相当大的比重。此外，某些改造过的批任务由于频繁对特定资源进行锁定，也会对独立事务造成相当大的影响。如果在设定时间内，数据库服务器由于资源锁定没有能够完成客户端发出的操作请求，数据库服务器将通知被锁定的客户端该操作超时。

某些大粒度事务在并发执行的实例较多时也会造成同类或不同事务的数据库超时。

此外，应用系统如果没有健壮的异常处理机制，很可能造成锁资源不被释放（即，开始的事务既没有提交也没有回滚）。当这种错误发生时，必然造成资源被长久锁定。对此类问题，应用系统在开发的过程中需要采取一套完善的异常处理机制，确保资源不被长期锁定。

- 数据库死锁。

由于数据库死锁可以看作进程间死锁的一种特殊情况，我们可以采取与处理操作系统死锁相类似的方法解决数据库死锁的问题。造成死锁必须具备下述条件（Coffman et al 1971）：

① 互斥条件。每一个资源或者被分配给特定的进程，或者可用。

② 持有并等待条件。被授权持有资源较早的进程可以请求新的资源。

③ 不可取代原则。事先被赋予的资源不能够从该进程被强制取走，它们必须被所持有的进程明确释放。

④ 环等待条件。必须存在两个或者多个进程的环形链，其中每一进程都等待由环形链的下一个成员所持有的资源。

由于软件开发人员对资源争用可能造成的死锁问题往往没有充分考虑，而目前主流数据库管理系统主要采用乐观的并发控制算法，导致应用系统实际使用过程中频繁发生死锁。应该说明，同类型数据库事务的不同实例之间由于访问资源的顺序一致，通常情况不会发生死锁；不同类型事务之间如果没有按照一个统一的契约进行并发访问，将极容易形成死锁。因此，在确保应用系统功能的前提下，制定一个不同事务之间进行并发访问的原则，就可以有效消除环等待，减少死锁发生的可能性。

针对数据库的性能问题，一般应采用什么样的解决办法呢？

在对数据库服务器常见性能问题进行充分研究的基础上，我们制定了一套适用于解决已发布系统性能问题的通用方法，步骤如下：

① 监视性能相关数据；

② 定位资源占用较大的事务并做出必要的优化或调整；

③ 定位锁冲突，修改锁冲突发生严重的应用逻辑；

④ 对规模较大的数据或者无法通过一般优化解决的锁冲突进行分布。

必须指出，解决数据库性能问题是一个迭代和往复的过程，通常需要在各种条件的矛盾之间寻求最佳的平衡点。

（4）监视并记录性能相关数据

对数据库服务器软件、操作系统、网络环境乃至客户端等各类处理单元的性能相关信息进行监视并记录，是发现数据库性能问题的基础。这一步骤的作用是搜集与数据库服务器性能表现密切相关的数据，作为分析性能问题的基础。由于各个处理单元的状态是随着时间的推移而动态变化的，性能数据的监视与采集必须尽可能详细地记录下所有时间点上各个处理单元的状态信息。为此，我们采取对各个采样时间点的处理单元状态信息进行快照方式，来对性能相关数据进行监控和记录，相邻采样时间点之间的间隔越小，状态信息就越准确。

在各类监视活动中，对数据库服务器软件性能属性的监视是整个活动的重点，主要集中在数据库会话的状态信息、执行的结构化查询语句和锁使用情况等方面。其中，状态信息代表了单一数据库会话在其生命周期中的状态变化情况，包括在哪一个时间点开始一个事务，在哪一个时间点被其他会话锁定，何时超时等。执行的结构化查询语句代表单一数据库会话在其生命周期中执行的所有数据库操作。锁使用情况代表整个数据库服务器的锁资源使用和变化情况。此外，顺序扫描、高代价查询等属性也是代表数据库服务器性能的重要数据。

（5）定位资源占用较大的事务并做出必要的优化或调整

通过对数据库锁使用情况和 SQL 语句的执行历史进行分析，可以发现一个事务同时占用大量数据库锁的应用逻辑事务。通常这类事务都属于批任务。由于批任务本身的特性，决定了在其整个执行过程中，必然消耗大量资源，最好将其放置在系统具有充分空闲时间时进行。

（6）定位锁冲突，修改锁冲突发生严重的应用逻辑

如果应用系统锁冲突频繁发生，那么该系统的性能表现不可能令人满意。导致这种问题的原因非常复杂，主要表现在事务粒度过大、响应时间过长、异类事务互相影响并形成死锁等情况。通过对数据库锁使用情况信息的分析，可以定位发生锁冲突的各个会话；在此基础上对发生锁冲突的会话各自的执行状态变化和结构化查询语句进行分析，可以定位发生锁冲突的应用逻辑源程序。如果造成锁冲突的是同种或者异种普通事务，必须对其本身特性加以分析，确定是否本身粒度过大，数据访问是否存在瓶颈等。对这类事务的优化相对较难，一般需要开发人员的经验和对应用逻辑本身特性的了解。

（7）进行必要的数据分布

数据分布的主要目的是，通过数据库服务器的并行执行特性，使得单一事务的执行具有较短的响应时间和不同类的事务之间影响相对缩小。在缩短响应时间方面，这种方



法主要适用于对规模较大的数据库表进行访问的情况。它不仅使得特定的查询可以并行执行，而且有可能改变结构化查询语句的执行计划，缩小查询进行的范围。此外对于异类事务之间，或者同类事务的不同实例锁冲突频繁的问题，可以通过数据分布加以解决。

数据库性能问题通常表现在响应时间过长，并发处理能力差和锁冲突严重等方面，其原因是多方面的。本书提出的通过监视并记录应用系统处理单元性能相关数据，来定位性能问题的方法，可以帮助开发人员有效发现系统中存在的主要性能问题。必须指出，解决数据库性能问题是一个迭代和往复的过程，通常需要在各种条件的矛盾之间寻求合理的平衡点。

5. Oracle 与提高性能有关的特性

下面将以 Oracle 为例，讨论数据库优化的一些方法，对于其他的数据库，由于其实现的机制以及特性可能与 Oracle 不同，那么优化的方法也会有所不同，大家可以参照其手册，进行分析。

虽然这些方法属于数据库开发人员或者数据库管理人员优化系统的方法，但是如果测试人员了解这些方法，就可以更好地分析、定位数据库性能问题，制定有针对性的测试用例。

Oracle 与提高性能有关的特性主要包括：索引、并行执行、簇与散列簇、分区、多线程服务器以及同时读取多块数据等。下面分别进行介绍。

这里列出了 Oracle 配置的关键参数以及其使用方法。

① max_dspatchers：指定了系统允许同时进行的调度进程的最大数量。

② max_shared_servers：指定了系统允许同时进行的共享服务器进程的最大数量。如果系统中出现的人为死锁过于频繁，那么管理员应该增大这个参数的值。

③ parallel_adaptive_multi_user：当该参数的值为 true 时，系统将启动一个能提高使用并行执行的多用户系统性能的自适应算法。这个算法将根据查询开始时的系统负载自动降低查询请求的并行度。

④ parallel_automatic_enabled：如果将该参数的值设置为 true，那么 Oracle 将确定控制并行执行的参数的默认值。

⑤ parllel_broadcast_enabled：该参数允许管理员提高散列连接和合并连接操作的性能，在这样的连接操作中，系统将一个大尺寸的结果集与一个小尺寸的结果集连接在一起（在合并操作中，数据的尺寸是根据字节数而不是记录数确定的）。

⑥ parllel_execution_message_size：这个参数指定了系统并行执行时的消息尺寸（在 Oracle 的旧版本中，这个概念是指并行查询、PDML、并行恢复和并行复制数据等）。

⑦ parllel_max_servers：指定了实例能同时运行的并行执行进程和并行恢复进程的

最大数量。随着用户需求的增长，在创建实例时，为这个参数设置的值将不再能满足用户需求，所以应当增大这个参数的值。

⑧ parllel_min_percent：系统将联合使用 parllel_max_servers、parllel_min_servers 和该参数。这个参数允许指定并行执行进程（即参数 parllel_max_servers 之值）的最小百分比。

⑨ parllel_min_servers：这个参数指定了实例并行执行进程的最小数量。其值就是实例启动时 Oracle 创建的并行执行进程数。

⑩ parllel_threads_per_cpu：指定了实例默认的并行度和并行自适应以及负载平衡算法。它指明了并行执行过程中一个 CPU 能处理的进程或线程数。

⑪ partition_view_enabled：指定了优化器是否使用分区视图。Oracle 推荐用户使用分区表（这是在 Oracle8 之后引入的）而不是分区视图。分区视图只是为了提供 Oracle 的后向兼容性。

⑫ revovery_parallelism：这个参数指定了恢复数据库系统时使用的进程数。

（1）索引

在数据库系统中，索引是一种可选结构，其目的是提高数据访问速度。利用索引可提高用户访问数据的速度，或直接从索引中独立检索数据。如果对索引的配置和使用进行了优化，那么索引能大大降低数据文件的 I/O 操作并提高系统性能。

但是在为一个表创建索引之后，Oracle 将自动维护这个索引。当用户在表中插入、更新或删除记录时，系统将自动更新与该表相关的索引。一个表可以有任意数量的索引，但一个表的索引越多，用户在该表中插入、更新或删除记录时所造成的系统开销也越大。其原因是无论何时更新表，系统都必须更新与之相关的索引。

索引是建立在表的一个或多个字段之上的。索引的作用大小取决于该字段或字段集的选择性。所谓选择性，是指索引能降低数据集中的程度。如果表中与某个索引相关的字段值各不相同，那么该索引就有很好的选择性。一个选择性很差的索引的例子，是基于字段值仅为 true/false 的字段创建的索引，因为表中很多记录该字段的字段值都相同。一个索引可能只能帮助管理员降低检索的记录数，而不能惟一地确定一条记录。例如：如果为一个表的 LastName 字段创建了一个索引，现在用户需要搜索 John Smith，那么这个索引将返回 LastName 字段值为 Smith 的所有记录，因而用户还不得不在返回的记录中搜索含 John 的记录。索引的选择性越好，就越有助于降低返回记录的数量，从而提高数据访问速度。下面介绍有效创建和使用索引的技巧和方法。

- 索引和降低系统处理的数据量。

索引的主要作用之一就是降低系统处理的数据量。对 CPU 使用和等待完成 I/O 操作的时间上，I/O 操作引起的系统开销都是非常昂贵的。降低 I/O 操作可提高系统性能和处



理能力。如果不使用索引，那么为了找到特定的数据，系统将不得不扫描表中的所有数据。

例如如下查询语句：

```
SELECT *
  FROM emp
  WHERE employee_id=10290
```

如果不使用索引，系统必须扫描整个 emp 表并检查表中每条记录的 employee_id 字段的值。如果 emp 表很大，那么这个操作可能意味着数量巨大的 I/O 读写和很长的处理时间。

如果为 emp 表的 employee_id 字段创建了索引，那么系统将遍历该索引并找到用户所查询记录的 ID。找到记录 ID 之后，只需一条额外的 I/O 操作就能检索到用户所需的数据。

用于说明这个问题的最好例子，是只需查找一条记录的情况。在表的每条记录中，类似 employee_id 这样的字段的值可能在整个表中都是惟一的。这意味着查询结果值返回一条记录，这种查询的效率是非常高的。

在某些情况下，索引必须返回大量数据。如下面的例子：

```
SELECT *
FROM customers
WHERE region='North'
```

这个查询语句很可能返回大量数据，因为索引操作返回了大量记录的 ID，并且系统必须独立访问这些记录的 ID，所以这种情况下，不使用索引可能比使用索引的效率更高，直接进行表扫描可能效率更高。不同情况下，采用哪种查寻方法更好，很大程度上取决于表的数据量和组织形式。

对于不同的数据，在某些情况下位图索引可能非常有用，而在另外一些情况下，使用位图索引可能没有任何好处。

- 索引和更新。

如果对表创建了索引，那么更新、插入和删除表中的记录都将导致额外的系统开销。在系统提交这些操作之前，系统将会更新所有与该表相关的索引。这可能需要花费很长时间，并额外增加一定的系统开销。

- 在字段选择性很低的情况下适用索引。

在某些情况下，表中的某些字段的选择性可能很低。开发人员没必要为所有表创建索引，实事上，在某些情况下索引引起的问题比解决的问题更多。在很多情况下，需要

反复试验，才能确定一个索引是否有助于提高系统性能。

但是，位图索引能在字段选择性不高的情况下工作得很好。一个位图索引可以和其他位图索引联合使用，以降低系统检索的数据集。对于某些值为 true/false、yes/no 或其他小范围数据的字段，建立位图索引是非常合适的。请记住：位图索引所占用的空间，是随着与该索引相关的字段的不同值的数量的增加而增加的。

如果决定创建一个索引，那么确定为哪些字段创建索引是非常重要的。对于不同的表，可能会选择一个或多个字段创建索引。可使用如下方法来确定在哪些字段上创建索引：

① 选择那些最常出现在 where 子句中的字段。经常被访问的字段最可能受益于索引。

② 经常用于连接表的字段是创建索引的必然候选字段。

③ 必须注意索引导致的查询语句性能的提高与更新数据时性能的降低之间的平衡。

④ 经常被修改的字段不适合创建索引，其原因是，更新索引将增加系统开销。

在某些情况下，使用复合索引的效率可能比使用简单索引的效率更高。下面的一些例子说明了应当在何种情况下使用复合索引。

① 某两个字段单独来看都不具有惟一性，但结合在一起却有惟一性，那么这种情况下，复合索引将工作得很好。例如：A 字段和 B 字段都几乎没有惟一性值，但绝大多数情况下，字段 A 和 B 的某个特定组合却具有惟一性特点。那么在检索数据时，可在 where 子句重视 and 操作符来将这两个字段连接在一起。

② 如果 select 语句中的所有值都位于复合索引中，那么 Oracle 将不会检索表，而直接从索引中返回数据。

③ 如果多个查询语句的 where 子句中作为查询条件的字段都不相同，但返回的记录相同，那么应当考虑利用这些字段创建一个复合索引。

在创建索引之后，开发人员应当定期利用 SQL TRACE 工具或 EXPLAIN PLAN 来察看用户查询是否充分利用了索引。很有必要花费一定精力来试验使用索引和未使用索引在效率上的差别，以判断索引所耗费资源是否物有所值。

应该删除那些不经常使用的索引。可使用 alter index monitoring usage 语句来跟踪索引的使用情况。还可以从系统表 all_indexes、user_indexes 和 dba_indexes 中查询用户访问索引的频率。

如果为一个不适合创建索引的字段或表创建了索引，那么这可能会导致系统能力的下降。而如果创建的索引合理，那么这将降低系统的 I/O 操作并加快访问速度，从而大大提高系统性能。

（2）Oracle 的并行执行特性

Oracle9i 的并行执行特性就是 Oracle 旧版本中读者熟知的并行查询选项。而在

Oracle9i 中，这个特性已内嵌在 Oracle RDBMS 中。

为什么要实现并行执行？RDBMS 的绝大多数操作都可分为以下 3 类：

① 被 CPU 限制的操作：这类操作的速度和单 CPU 运行速度一样。通过并行化操作，多个 CPU 可并行处理系统负载，因此可以更快完成该操作。

② 被 I/O 限制的操作：这类操作花费了绝大部分时间等待系统完成 I/O 操作。当系统中同时出现多个 I/O 请求时，绝大多数 raid 控制器将很好地工作。另外，当一个线程需要等待完成 I/O 操作时，可充分利用 CPU 来处理另一线程的 CPU 部分。

③ 被竞争限制的操作：并行处理不能改善由资源竞争所限制的操作。

对表扫描而言，系统花费在等待数据从磁盘返回的时间常常比处理数据所花费的时间还长。通过并行机制，可用多个服务器进程处理查询操作，从而弥补了系统在这方面的问题。当一个进程在等待 I/O 操作时，CPU 可执行另一个进程。如果数据库系统运行在对称多处理器（Symmetric Multiprocessor，SMP）计算机、计算机群集或大规模并行处理（Massive Parallel Processing，MMP）计算机上，那么开发人员可充分利用并行处理机制的优势。

并行执行使 Oracle 的某些功能由多个服务器进程协同处理成为可能。这些功能包括查询、创建索引、加载数据和恢复数据库。所有这些功能都遵循一个共同规则，即充分利用 CPU 资源。

- 并行查询处理。

并行查询处理允许多个服务器进程以并行方式处理某些 Oracle 语句。

并行查询操作将该查询分为几个不同的部分，每部分由不同的服务器进程进行处理。这些进程称为查询服务器。一个被称为“查询协调者”的进程负责调度这些查询服务器。系统将一个 SQL 语句和一个并行度提交给查询协调进程，而协调进程则负责将该查询分配给查询服务器，并将每个查询服务器返回的结果组合成一个整体。并行度是指分配给该查询的查询服务器数量。Oracle 服务器能对连续、排序以及表扫描操作实现并行执行。

在多处理器或并行处理计算机上，并行查询操作是非常有效的；在单处理器计算机上，如果大部分时间都花在了等待系统完成 I/O 操作上，那么使用并行查询也是非常有效的。具有足够 I/O 带宽的系统，尤其是使用磁盘阵列的系统，都能够从并行查询操作中受益。

如果通常情况下系统 CPU 的使用率达到了 100%，并且系统中只有少量磁盘驱动器，那么使用并行查询是没什么意义的。同样，如果系统的内存非常紧张，那么并行查询也没有多大意义。

在并行查询中，合理的 I/O 分布和并行度是最需要调整的两个方面。对并行度的调

整一方面需要反复的试验，另一方面还需要一定的分析。应当首先根据如下一些因素考虑并行度。

① 计算机的 CPU 能力：CPU 的数量和能力将影响系统能运行的查询进程数量。

② 系统处理大量进程的能力：一些操作系统能处理很多并发进程，而另一些操作系统则没有这方面的能力。

③ 系统负载：如果系统现在的运行已经达到了极限，那么对并行度的调整不会有太大效果。如果系统运行已达其能力极限的 90%，那么太多的查询进程将使系统不堪重负。

④ 系统处理的查询数量：如果系统的大部分操作是更新操作，但仍有少量的重要查询存在，那么开发人员可能希望系统运行多个查询进程。

⑤ 系统的 I/O 能力：如果磁盘上的数据是分片或是使用磁盘阵列存储的，那么系统能够处理多个并行查询。

⑥ 操作类型：系统是否需要处理很多的全表扫描或排序？并行查询服务器非常有助于这类操作。

这些因素对系统采用的查询并行度有一定影响。下面是关于并行度的另一些建议。

① 诸如排序之类的，需要大量 CPU 资源的操作应当采用较低的并行度。其原因是这类受限于 CPU 的操作已经充分利用了 CPU，而不需等待系统的 I/O 操作。

② 诸如全表扫描之类的，需要大量磁盘 I/O 的操作，应当采用较高的并行度。需要等待磁盘 I/O 的操作越多，系统就越能受益于并行操作。

③ 如果系统中有大量的并发进程，那么应当采用较低的并行度。因为太多的进程将使系统不堪重负。

在确定使用并行操作之后，读者可通过查询动态性能视图 V$PQ_SYSSTAT 来监控系统。

- 并行创建索引。

并行查询的另一个特征是以并行方式创建索引的能力。通过这个特性，系统用于创建索引的时间将大大减少。类似于并行查询操作，并行创建索引操作也有两组查询服务器。其中一组查询服务器的功能是扫描需要创建索引的表，以获得 ROWID 和与索引相关的字段值；另一组查询服务器则对第一组查询服务器所得到的值进行排序，并将排序后的结果传递给协调进程。而协调进程再根据这些排序之后的条目创建 B 树索引。

- 并行加载数据。

通过使用多个并发会话同时往同一个表中写数据，可实现数据加载的并行化。对于不同的系统配置，通过并行方式加载数据，可以提高数据加载性能。因为加载数据不仅



需要大量的 CPU 资源而且还需要大量的 I/O 操作，所以并行化数据加载将提高数据加载的性能。并行化数据加载将通过多个直接加载器进程实现。

并行加载数据是非常有用的，特别是在数据加载时间很紧张的环境中尤其如此。通过将每个输入文件放置在不同的卷上，可提高并行加载数据的性能。并行查询操作中所有的通用调整原则都是用于并行加载数据操作的。

- 并行恢复。

并行恢复是并行查询选项的一个非常重要的特性。在评估 Oracle 和测试软硬件时，需要有意地使系统崩溃，以检验系统的可恢复性。通过使用并行恢复选项，恢复数据库和实例的时间将大大缩短。当被恢复的系统有很多磁盘并支持异步 I/O 操作时，系统恢复所用的时间将会大大缩短。对于只有很少磁盘的小系统或不支持异步 I/O 的系统，采用并行方式恢复系统是不明智的。在并行恢复方式中，一个进程负责从重做日志中读取和调度重做入口，并将这些入口传递给恢复进程。而恢复进程则负责将用户对数据库所作的修改写进数据库中。

总之，在平衡操作负载方面并行查询选项是非常有用的，当其他进程在等待系统完成 I/O 操作时，并行查询可使 CPU 转而处理其他进程，从而充分利用系统的 CPU 资源。对于多处理器计算机，使用并行查询是非常有用的，但这并不是说并行查询不能用于单处理器计算机。

（3）簇

簇（cluster），有时被称为索引簇，是 Oracle 数据库中用于存储表的一种方法。在一个簇中，系统将多个相关的表存储在一起，以缩短用户访问相关记录的时间。只有当这些相关表经常被同时访问时，才适合使用簇。对用户和应用程序而言，簇的存在是透明的，簇只影响数据的存储方式。

在某些情况下使用簇是非常有利的，而在另外一些情况下，使用簇却可能非常不利。应当仔细考虑簇是否有助于提高系统性能。一般而言，如果集中存放的数据主要用于连接表中，那么使用簇是很好的。如果两个表存放了相关数据，并且这两个表经常被同时访问，那么通过使用簇可将相关数据预装入 SGA 中，从而提高用户访问数据的性能。因为开发人员经常同时使用这两个表，所以在用户访问其中一个表时，将另一个表的数据也放入 SGA 中，可大大缩短用户访问数据的时间。如果一般情况下开发人员不会同时使用这些信息，那么簇将不能提高系统性能，并且这种情况下，簇实际上会导致系统性能的轻微下降，其原因是额外的表信息将占据更多的 SGA 空间。簇的另一个不足之处在于，当用户执行 insert 语句时将降低系统性能。引起性能下降的原因是簇在使用存储空间上采用的方法更加复杂，并且系统需要将多个表存储在同一个数据块中。簇表比单个表占用了更多的存储空间，这将导致系统扫描更多的数据。另外如果系统经常对这

些表中的某一个表作全表扫描，那么不应当为这些表创建簇。因为如果创建了簇，那么额外数据将占用 SGA 的部分空间并导致额外的 I/O 操作，这两方面的原因都会降低系统性能。

（4）散列簇

散列簇和簇非常相似，但散列簇采用散列函数而不是索引访问簇键。散列簇根据散列函数的计算结果保存数据。散列函数是一个数学函数，这个函数根据散列键的值来确定簇中的数据块。

尽管可以以类似于索引簇的方式创建散列簇，但不必为所有表创建散列簇。实际上，更经常做的是为一个单独的表创建散列簇，以利用散列簇的特性。通过使用散列索引，系统只需一次 I/O 操作，系统就能检索到用户所需数据，如果使用 B 索引，系统需要多次 I/O 操作才能检索到需要的数据。适合创建散列簇的表应具有以下特征：

- 簇键值具有惟一性。
- 用户对表的大部分查询都是关于键值的等式查询。
- 表的尺寸是静态的，几乎不会出现任何增长。
- 簇的键值不会发生任何变化。

一个适合于创建散列簇的典型例子是一个用于存储零件信息的表。通过建立一个关于零件号的散列键值，用户对该表的访问将非常有效而且非常快。任何时候，如果系统中有一个静态的表，其中的一个或多个字段的值具有惟一性特征，那么可考虑为这样的表创建散列簇。

和索引簇一样，使用散列簇既有优点也有不足。当用户使用关于簇键的等式查询检索数据时，散列簇具有很高的效率。如果用户不是根据簇键值检索数据，那么这个查询就不能被散列化。这就像在索引簇中见到的那样，当在散列簇表中使用 insert 语句时，将降低系统性能。

（5）同时读取多块数据

当系统执行表扫描时，Oracle 具备同时读取多个数据块的能力，这种能力提高了系统的 I/O 速度。通过同时读取多块数据，Oracle 能够从磁盘上读取更大的数据块，从而避免了对磁盘上数据进行搜索的操作。通过降低磁盘搜索和读取更大的数据块，可以降低系统的 I/O 开销和 CPU 开销。

Oracle 的这个特性被称为多块读取。只有对于连续数据块，多块读取特性才有助于提高系统性能。通常情况下，一个盘区内的数据块都是连续的。如果数据存放在多个小盘区上，那么多块读取的性能将会降低。

Oracle 初始化参数 db-file-multiblock-read-count 指定了一次多块读取能读取的数据量。因为这个参数几乎不会导致系统性能的下降，所以一般情况下，都将这个参数的值



设置得很高。I/O 的尺寸取决于初始化参数 db-file-multiblock-read-count 和 db-block-size 的值。

为了充分发挥多块读取数据的优势，应当尽量配置自己的系统以使数据库的块尽可能都是连续的。为了做到这一点，应当使用优化尺寸的盘区来创建数据库。盘区的尺寸应当远大于一次多块读取操作所读取的数据的尺寸。

（6）分区

分区（partition）这一特性的目的是允许大尺寸表或索引分解成小的更容易管理的部分，即分区。与原来的表或索引相比，因为分区变小了，所以系统访问分区的速度更快、效率也更高，可将分区想象成表或索引被分解之后的多个小尺寸的表或索引。系统可单独访问这些小尺寸的表或索引，也可以以组或整体方式访问之。

可使用多种分区方案来创建分区。这些方案确定了如何将数据分解成分区。可使用如下一些方案。

- Range Partitioning：这种方案根据数据的范围，比如月、年等对表中的数据进行分区。
- List Partitioning：这种方案和 Range Partitioning 分区方案很类似，但这种方案是按照数据的值，而不是数据的范围来进行划分的。
- Hash Partitioning：这种分区方案使用散列函数来实现对数据的自动分区。
- Sub-Partitioning：这种方法就是开发人员熟悉的复合分区方法。这种方法允许开发人员同时使用多种分区方案。

分区有以下几方面的优点。

- 对能被分区的大尺寸表进行扫描时，分区可降低 I/O 操作和 CPU 的使用率。
- 可在分区的层次上而不是表的层次上加载数据。
- 能以删除分区的方式删除数据，而不必使用 select 语句来删除大量数据。
- 对用户和应用程序而言，分区是完全透明的。
- 可在分区层次上而不是在表层次上维护数据。

在很多环境中，这些好处能极大地提高系统性能。

（7）多线程服务器

用户可通过专用服务器进程连接到 Oracle 实例，也可以通过多线程服务器进程连接到 Oracle 实例。因为每一个专用服务器进程都将占用大量内存资源和系统资源，所以有必要对多用户连接采用多线程服务器进程。

多线程服务器进程允许多个用户使用一定数量的共享服务器进程。所有对共享服务器的请求都必须经过调度进程，这个过程对 SGA 大缓冲池中的用户对共享服务器进程的请求进行排队。当系统处理完用户请求后，系统通过 SGA 中的大缓冲池将请求结果

返回给调度进程。

因为共享服务器进程使用共享缓冲池对用户请求进行排队并返回数据，从而大大减少 CPU 和内存的使用。然而，正如读者所猜想的那样，多线程服务器也会增加系统开销。这就是为什么执行需要很长时间的批处理作业时，使用专用服务器的原因。

对配置和调整多线程服务器，需要调整参数文件中的参数。还应当小心监控系统的大缓冲池，以确保系统的运行没有超出分配给它的内存空间。

应当尽量监控只有少量用户的共享会话所使用的内存。监控结果将会告知共享会话使用了多少内存。利用这些数据，就能估算所有会话总共需要多少内存。可通过下面的 SQL 语句获得这些信息：

```
SELECT SUM(value) || 'bytes' "Memory"
FROM v$sesstat,v$statname
WHERE name ='session memory'
AND v$sesstat.statistic#=v$statname.statistic#;
```

上面语句的输出结果告诉开发人员使用了多少内存。将使用的总内存量除以连接数，就可确定每个会话使用的内存数量。然后就可从每个会话使用的内存量来推算为了支持系统的所有会话，需要为系统的大缓冲池分配多大内存空间。

如果认为大缓冲池的内存太少，那么可通过调整初始化参数 large_pool-size 来增加大缓冲池的内存空间。请记住：除多线程服务器之外，系统还使用大缓冲池来执行并行查询操作。

初始化参数 mts_dispatchers 决定了每个网络协议使用的调度进程的数量。通过提高这个参数的值，可能会看到用户会话的性能得到了提高，其原因是用户会话不再需要等待可用调度进程。

除前面讨论的参数之外，还有如下一些参数与多线程服务器有关。

- mts_max-dispatcher：实例能创建的调度进程的最大数量。这里的调度进程包括所有网络协议的调度进程。
- mts_servers：实例刚启动时的共享服务器进程数。如果将这个参数设置为 0，那么 Oracle 将不会是用共享服务器进程。为了满足系统需要，共享服务器进程的数量将会动态增长。
- mtx_max_servers：这个参数指定了实例中允许运行的共享服务器进程的最大数量。

如果应用程序能利用簇和索引的优势，那么簇和索引将有助于提高系统性能。如果应用程序使用了索引而又需要频繁更新数据，那么这样的索引将会降低系统性能，其原



因是在数据更新的同时，系统也需要更新索引。同样地，如果用户的数据访问模式使用了簇，那么簇将会有助于提高系统性能；反之，簇将会成为系统的负担。

Oracle 并行查询选项和并行服务器选项能极大地提高系统性能。通过将查询分配成多个服务器进程，并行查询选项提供了一种提高系统 CPU 使用率的机制。并行服务器为开发人员提供了一个健壮的可升级的服务器群集功能，这个特性不仅能提高系统性能，还为系统提供了立即排除错误的能力。应用程序和需求将决定这些特性是否有助于提高系统性能。在调整系统时，非常重要的一点是需要注意调整系统所冒的风险。在前面的内容介绍中，我们讨论了不同的系统调整主体，例如共享内存池的尺寸和数据块缓冲区等。提高共享内存池和数据库缓冲区的尺寸将会提高缓冲区的命中率，但这样的调整也存在着损害系统性能的风险。然而，如果读者对某些系统特性配置得不正确，那么它们对系统性能造成的负面影响会更大。任何时候调整系统，都应当预先估计该调整所面临的风险，特别是在没经过详细测试就对生产系统进行调整时更是如此。

如果对系统需要调整的方面和该调整所影响的方面进行了仔细考虑，并对应用程序的数据访问模式有深入的了解，那么可以将调整面临的风险降到最小。应当仔细考虑对系统所作的修改将会对系统造成的影响，并尽量了解系统修改对应用程序的影响。只有这样，才能在优化系统性能的同时，将系统调整的风险降到最低。

6．测试报告

一般情况下，在测试的标志性阶段或者测试结束时需要出具测试报告，测试报告是整个测试的总结，其主要作用是描述测试结果。测试报告的格式可以不拘一格，但需要验证是否包括以下关键内容：

- 测试案例说明；
- 测试结果数据；
- 测试结果分析；
- 测试环境说明；
- 报告术语解释。

这些内容在前面都有过详细论述，这里不再赘述。

8.5 负载压力测试技巧

8.5.1 参数池技术

录制业务流程时，测试工具生成一个由函数构成的 Vuser 脚本。函数中参数的值是录制期间使用的实际值。例如，假设在操作 Web 应用程序时录制了一个 Vuser 脚本。

VuGen 生成下列语句，在库的数据库中搜索标题“UNIX”，使用多个 Vuser 和迭代来重播该脚本时，如果不想重复使用相同的值“UNIX”，那么可以用参数来替换该常量值。

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library. TITLE",
        "value=UNIX",
        ENDITEM,
        "name=library. AUTHOR",
        "value=",
        ENDITEM,
        "name=library. SUBJECT",
        "value=",
        ENDITEM,
        LAST);
;

web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library. TITLE",
        "value=(Book_Title)",
        ENDITEM,
        "name=library. AUTHOR",
        "value=",
        ENDITEM,
        "name=library. SUBJECT",
        "value=",
        ENDITEM,
        LAST);
```

上例中可以用参数“Book_Title”来替换常量值“UNIX”，然后，生成的 Vuser 使用指定的数据源中的值来替换参数。该数据源可以是一个文件或者内部生成的变量。

对 Vuser 脚本进行参数化有两个好处：

- 减小脚本的大小。
- 提供了使用不同的值测试脚本的能力。例如，如果要在数据库中搜索几个标题，只须写一次提交函数，在函数中使用参数，而不是指示 Vuser 搜索一个特定的标题。重播期间，VuGen 用不同的值替换该参数。

参数化涉及下列两个任务：



- 用参数替换 Vuser 脚本中的常量值；
- 为参数设置属性和数据源。

下面谈谈数据源的问题。可以选择一个文件作为参数值的源，一个常见的使用参数的方法是指示 Vuser 从外部文件中取值。可以执行下列步骤。

- 选择或者创建数据文件；
- 设置参数的属性。

还可以从现有数据库中导入数据，用于参数化。可以用下列两种方法中的一种导入数据。

- 新建查询；
- 指定 SQL 语句。

8.5.2 将事务插入到 Vuser 脚本

可以定义事务以度量服务器的性能。每个事务度量服务器响应指定的 Vuser 请求所用的时间。这些请求可以是简单任务（例如等待对单个查询的响应），也可以是复杂任务（例如提交多个查询和生成报告）。要度量事务，需要插入 Vuser 函数以标记任务的开始和结束。在脚本内，可以标记的事务不受数量限制，每个事务的名称都不同。在方案执行期间，主控台将度量执行每个事务所用的时间。场景运行后，可使用测试工具的图和报告来分析各个事务的服务器性能。

8.5.3 将集合点插入到 Vuser 脚本

要在系统上模拟较重的用户负载，需要同步各个 Vuser 以便在同一时刻执行任务。通过创建集合点，可以确保多个 Vuser 同时执行操作。当某个 Vuser 到达该集合点时，主控台会将其保留，直到参与该集合的全部 Vuser 都到达。当满足集合条件时，主控台将释放 Vuser。

可通过将集合点插入到 Vuser 脚本来指定会合位置。在 Vuser 执行脚本并遇到集合点时，脚本将暂停执行，Vuser 将等待主控台允许继续执行命令。

8.5.4 手工关联

这一小节我们主要解决动态数据所导致的问题，即利用测试工具的脚本函数如何关联动态且不可人工预知的值。

系统的输出值需要为后续操作提供输入，这些值只对当前会话有效。举例说明如下。

- 系统产生的 SessionID；
- 每次访问 Web 页面的动态 URL；

- 表单提交期间录制的 Field（有时会隐藏）。

解决办法如下。

- 从一个操作步骤中捕捉输出值；
- 该值用于另一个步骤的输入。

关联数据先是由服务器发给客户端，之后客户端又会将该数据返回服务器。例如 SessionID，它是脚本中的一段代码，为录制的会话服务，但却不能用于回放的会话。

下面提供几种关联数据的方法。

- 手工关联；
- 录制结束后自动关联；
- 录制过程中自动关联。

在 Vuser 脚本中关联动态数据的步骤如下。

- 确定需要捕捉的值。

① 创建两个虚拟用户。这两个用户的录制步骤保持一致。如果说捕捉的动态数据依赖于某个输入值，那么就改变这个输入值；如果独立于任何输入值，那么就采用相同的数据。

② 对比脚本。利用 Wdiff.exe 工具对比，它遍历脚本的每一行，并且亮显不同点。

- 找到所捕捉值的左右边界标识符。
- 决定应该使用哪个边界。
- 将函数 web_reg_save_param 加入脚本，在加入之前，必需要先捕捉到值。
- 在函数中加入参数名称、左边界标识符、右边界标识符及函数事件。
- 在每次脚本运行时参数化动态数据。
- 校验执行结果。

8.5.5　IP 数据池

运行方案时，每台负载生成器计算机上的 Vuser 都使用该计算机的 IP 地址。也可以在负载生成器计算机上定义多个 IP 地址，以模拟用户使用不同计算机的真实情况。应用程序服务器使用 IP 地址来标识客户端。应用程序服务器经常缓存来自同一台计算机上的客户端信息。网络路由器则缓存源信息和目标信息，以提高处理能力。如果许多用户使用同一个 IP 地址，服务器和路由器都会进行优化处理。由于同一台负载生成器计算机上的 Vuser 具有相同的 IP 地址，服务器和路由器将进行优化处理，因而无法反映真实的情况。

测试工具的多 IP 地址功能可以使用许多 IP 地址来标识在一台计算机上运行的多个 Vuser。这样，服务器和路由器认为 Vuser 来自不同的计算机，因此使测试环境更加真实。



在 Windows 平台上，每块网卡上可以模拟的最大 IP 地址数为 35 个，Solaris（2.5.1 版）最多为 255 个，而 Solaris（2.6 或更高版本）最多则为 8192 个。

多 IP 地址功能适用于下列协议。

- 客户端服务器：DNS、Windows Sockets。
- 自定义：Java Vuser、Javascript Vuser、VB Vuser、VB Script Vuser。
- 电子商务：FTP、Palm、SOAP、Web（HTTP/HTML）协议、WinSock\WebDual 协议。
- bom：Oracle NCA、Siebel-Web。
- 邮件服务：Internet Messaging (IMAP)、MS Exchange (MAPI)、POP3、SMTP。
- 流数据：Real。
- 无线：i-Mode、VoiceXML、WAP。

以下过程说明了如何向负载生成器中添加新的 IP 地址。

- 运行负载生成器上的“IP 向导”添加指定数量的 IP 地址。为 UNIX 负载生成器计算机手动配置新的 IP 地址。
- 重新启动计算机。
- 如有必要，用新地址来更新服务器的路由表。
- 在控制台中启用这项功能。

8.5.6 Web 站点经验点滴

对 Web 站点的测试，从测试工具的角度来讲，给读者介绍下面几点经验。读者通过阅读本小节的内容应能达到触类旁通，将这些经验应用到其他的测试领域。

- 在执行客户端并发性能测试的过程中，需要同时监控数据库服务器、Web 服务器以及网络资源等使用情况，以便对系统的性能做全面评估。
- 录制的脚本需要编辑，有时需要手工编写脚本。
- 尽可能去录制脚本，然后在其基础上编辑脚本。
- 手工编写脚本需要注意既能够模拟负载压力，又符合脚本的后台处理方式。
- 设置数据池，实现变量替换常量。为了真实模拟负载，数据池是经常使用的有效手段。
- 混合业务批量执行。

单独的业务并发操作，有可能会忽略例如资源争用、锁冲突等问题，在 Web 站点负载压力测试方案中，一定要考虑将多种业务混合执行，并发性能测试。

- 模拟用户数的递增。

我们知道在真实情况下，高峰期负载压力的到来是循序渐进的过程，同样的道理，

高峰期的结束也有一个过程。在工具中我们使用虚拟用户数的递增与递减来模拟这种情况。

- 合理设置交易之间的时间间隔。

交易之间的时间间隔代表了负载程度的高低，为了模拟不同的负载，经常需要调整此时间间隔。

- 模拟 IP 地址变量的技术。

并发访问需求量不大的系统，每个不同的虚拟用户使用不同的 IP 地址访问服务器是非常有必要的。

- 超时（timeout）的设置。

这项设置与系统 Web 服务器、数据库服务器、中间件服务器等超时设置有关，建议工具的设置值大于等于系统服务器的设置值。

- 并发用户连续执行交易数的设置。

每个虚拟用户在并发时，串行循环执行的交易数建议设置为 3～5 个。

- 错误跟踪。

测试期间的报错是故障定位的主要依据，应该分清错误的来源，包括服务器端错误、客户端错误以及网络错误。

- 利用动态数据处理技术。

对某些动态值，每次执行它都在变化，如果不加处理，往往导致负载测试失败。

- 尽量将执行负载测试的机器合理分布。

将负载生成器布置在不同的网段，有利于模拟来自不同用户群的负载。

- 并发用户数量极限点。

压力测试的目的是测试系统能够支持的最大并发用户数。

- 负载生成器的资源使用率也有必要监控。

负载生成器的资源使用超出范围，导致模拟客户端并发请求失败。

- 设置并发集合点。

在脚本中设置并发集合点，可以将录制的完整操作过程分解为一个个小的并发交易。

- 工具参数的配置。

工具参数的配置非常灵活且有效，在下面的章节中将详细论述。

8.5.7 脚本调试技术

下面介绍几个脚本调试的例子，读者可在此基础上进一步提高调试脚本的技术。

1．TUXEDO

这一类中间件，脚本处理的重点包括：



- 管理 TUXEDO 缓存。
- 在 TUXEDO 命令之间传输数据。

2．Winsock

Winsock 脚本调试的重点是在脚本中如何用变量来代替定值，即处理 Winsock 应用程序数据流。我们来看下面的示例脚本。可参见 http://yuanhaisong.myip.org/sqa/webtest.htm。

原始脚本如下：

```
PLAYER_INFO *s_info;
{
//声明变量
SET_ABORT_FUNCTION(abort_function);
DEFINE_TRANS_TYPE("wsk-AdvancedTech_1.c");
DefaultCheckpointsOn();
DO_WSK_Init(s_info);
SetTimeout(20);
SYNCHRONIZE();
BEGIN_TRANSACTION();
DO_WSK_Socket(S1, AF_INET, SOCK_STREAM, IPPROTO_IP);
DO_WSK_Bind(S1, ANY_ADDR, ANY_PORT);
DO_WSK_Connect(S1, "172.22.24.125", 2100, AF_INET);
/////////////////////////////////////////////////////////
// 服务器返回的 Session ID 在每次连接时是惟一的
/////////////////////////////////////////////////////////
* 21bytes: SessionID=jrt90847\r\n */
DO_WSK_Expect(S1, "\n");
/////////////////////////////////////////////////////////
// 惟一的 id 用于接下来的请求
/////////////////////////////////////////////////////////
/* 34 bytes */
DO_WSK_Send(S1,
"SessionID=jrt90847\r\n:^B^@^@^@^B^@^@^@^A^@^@^@");
/* 15 bytes: ID Accepted#^@\r\n */
DO_WSK_Expect(S1, "\n");
DO_WSK_Closesocket(S1);
END_TRANSACTION();
REPORT(SUCCESS);
EXIT();
```

```
return(0);
}
```

修改后的脚本如下：

```
PLAYER_INFO *s_info;
{
//声明变量
char Buffer[64];
char SendBuffer[64];
int nBytesReceived = 0;
SET_ABORT_FUNCTION(abort_function);
DEFINE_TRANS_TYPE("wsk-AdvancedTech_1.c");
DefaultCheckpointsOn();
DO_WSK_Init(s_info);
SetTimeout(20);
SYNCHRONIZE();
BEGIN_TRANSACTION();
DO_WSK_Socket(S1, AF_INET, SOCK_STREAM, IPPROTO_IP);
DO_WSK_Bind(S1, ANY_ADDR, ANY_PORT);
DO_WSK_Connect(S1, "172.22.24.125", 2100, AF_INET);
/////////////////////////////////////////////////////
//服务器的响应读入 Buffer 变量，对于本次连接我们就取得了惟一的
Session ID. /////////////////////////////////////////////////////
DO_WSK_Recv(S1, Buffer, 64, 0, &nBytesReceived);
Buffer[nBytesRecieved] = '\0';
/* 21bytes: SessionID=jrt90847\r\n */
//DO_WSK_Expect(S1, "\n");
/////////////////////////////////////////////////////
// 最终用脚本中的一个编码取代了从服务器接收到的 Session ID
/////////////////////////////////////////////////////
sprintf(SendBuffer, "%s:^B^@^@^@^B^@^@^@^A^@^@^@",
Buffer);
DO_WSK_Send(S1, SendBuffer);
/* 34 bytes */
//DO_WSK_Send(S1,
"SessionID=jrt90847:^B^@^@^@^B^@^@^@^A^@^@^@");
/* 15 bytes: ID Accepted#^@\r\n */
DO_WSK_Expect(S1, "\n");
```




```
Advanced Scripting Techniques for WinSock 7-5
DO_WSK_Closesocket(S1);
END_TRANSACTION();
REPORT(SUCCESS);
EXIT();
return(0);
}
```

3. SQL Server

这一类数据库，脚本处理的重点如下。

- 从存储过程中捕获一个值。
- 利用检索到的值作为一个参数传递给存储过程。

我们来看下面的实例脚本。

存储过程定义如下：

```
create procedure inc_test_sp
(
@first_param int
)
as
begin
select second_param = @first_param + 1
end
```

脚本代码如下：

```
strcpy(sql_statement, /* >> 1 << */
"execute inc_test_sp @sample_param ='{01}' ");
DO_substr(sql_statement, 1, "100" );
BEGIN_CHECKPOINT(); /* #1: Stored Procedure */
DO_dbcmd(0, sql_statement);
DO_dbsqlexec( 0 );
while (DO_dbGetResults(0));
END_CHECKPOINT(25); /* #25: Stored Procedure */
```

第一步：加入必要的变量说明。

```
int rhobot_script(s_info)
PLAYER_INFO *s_info;
```

```
{
int rhobot_script(s_info)
PLAYER_INFO *s_info;
{
char szSecondParam[20]; /* 假设数字长度 < 20!! */
long iSecondParam;
```

第二步：调用存储过程。

调用存储过程，然后修改其返回值。

```
strcpy(sql_statement, /* >> 1 << */
"execute inc_test_sp @sample_param ='{01}' ");
DO_substr(sql_statement, 1, "100" );
BEGIN_CHECKPOINT(); /* #1: Stored Procedure */
DO_dbcmd(0, sql_statement);
DO_dbsqlexec(0);
DO_addResultVar( "second_param" ); /* 注意这是关键的一行，此处修改返回值*/
while (DO_dbGetResults(0));
END_CHECKPOINT(25); /* #1: Stored Procedure */
strcpy( szSecondParam, DO_getResultVar( "second_param" ) );
iOutputReqID = atoi( szSecondParam );
RR_printf("Second Param (string): %s", szSecondParam );
RR_printf("Second Param (int): %d", iSecondParam );
```

原始脚本代码：

```
strcpy(sql_statement,/* >> 2 << */
"execute use_inc_value_sp @inc_value ={01}");
DO_substr(sql_statement, 1, "101" );
BEGIN_CHECKPOINT(); /* #2: Stored Procedure */
DO_dbcmd(0, sql_statement);
DO_dbsqlexec(0);
while (DO_dbGetResults(0));
```

修改后的代码（使用字符串值）：

```
strcpy(sql_statement, /* >> 2 << */
"execute use_inc_value_sp @inc_value ={01}");
/*注意 szSecondParam 在 Part 1 中已经声明并且接收到返回值*/
```

```
DO_substr(sql_statement, 1, szSecondParam );
BEGIN_CHECKPOINT(); /* #2: Stored Procedure */
DO_dbcmd(0, sql_statement);
DO_dbsqlexec(0);
while (DO_dbGetResults(0));
```

8.5.8 测试工具配置技巧

在测试工具中有很多配置参数，不同的配置有可能会导致测试失败，也有可能产生不同的测试结果。根据实际负载压力需求正确地配置参数，就可以保证达到真实地模拟负载，并得出正确的测试结果。测试实施过程中，配置测试工具参数的方法可以参考工具的用户手册，配置参数的技巧就要靠测试工程师的经验和技术积累了。下面举例说明配置参数。

- Form Field Comments（Yes / No）：在脚本中是否给 Form Field 部分加注释；
- Anchors as Comments（Yes / No）：在脚本中是否给 Anchors 部分加注释；.
- Client Maps Comments（Yes / No）：在脚本中是否给 Client Maps 部分加注释；
- Debug Comments（Yes / No）：在脚本中是否给 Debug 部分加注释；
- Doc Title Verification（Yes / No）：脚本录制过程中是否校验文档 Title；
- Baud Rate Emulation（Yes / No）：在脚本回放过程中是否模拟不同的带宽进行回放，如果需要，标明回放的带宽数值；
- Encode DBCS Characters（Yes / No）：是否将 DBCS 字符编码；
- Cache（Yes / No）：在脚本回放过程中，是否模拟缓存；
- Dynamic Redirect（Yes / No）：在脚本回放过程中，是否支持动态重定向；
- Dynamic Cookies（Yes / No）：在脚本回放过程中，是否支持动态 Cookies；
- Process Subrequests（Yes / No）：在脚本回放过程中，是否支持进程子请求；
- Persistent Connections（Yes / No）：在脚本回放过程中，连接是否持久保持；
- Max Concurrent Connection：在脚本回放过程中，最大当前连接数，默认值为 4；
- Max Connection Retries：在脚本回放过程中，最大当前连接重试数，默认值为 4；
- Server Response Timeout：在脚本回放过程中，服务器响应超时限制，默认值为 120；
- HTTP Version Detection：录制时采用的 HTTP 版本，默认值为 Auto，既可以为 1.0 版本，也可为 2.0 版本，测试工具自动处理；
- ActiveData（Yes / No）：在脚本回放过程中，是否支持动态数据；
- IPSpoofing（Yes / No）：在脚本回放过程中，是否支持每个虚拟用户使用不同的

IP 实现并发；

- Streaming Media（Yes / No）：是否支持流媒体；
- Hostnames as IP Addresses（Yes / No）：是否支持使用 IP 地址标识主机；
- Strip All Cookies From Requests（Yes / No）：在脚本回放过程中，请求中是否包括 Cookies；
- Traffic Filters（Yes / No）：在脚本回放过程中，是否需要流量过滤。

上述参数，例如“Strip All Cookies From Requests”，决定测试过程的成败，比如有些系统将 Session ID，或者用户的登录信息放在 Cookies 中，如果不加载 Cookies，那么自然导致测试失败。又如“Server Response Timeout”，此值如果设置不合适，可能会使负载压力测试过程报错，并且这个错误是由测试工具本身导致的。“Persistent Connections”这项参数根据我们的测试需求来确定，如果我们侧重测试系统并发用户数，那么就不应该选择这项参数，而是采用间断连接；如果我们侧重测试系统交易响应时间，那么选择这项指标会使系统响应发挥最好。“Baud Rate Emulation”帮助测试工程师实现在不同的网络带宽下实施负载压力测试等。



第9章　Web应用测试

9.1　Web系统测试概述

9.1.1　Web系统的构成

随着Internet的快速增长以及Intranet/Extranet在各行业的广泛应用，Web已经对商业、工业、银行、财政、教育、政府和娱乐等行业，以及我们的工作和生活产生了深远的影响。许多传统的信息和数据库系统正在被移植到互联网上，电子商务迅速增长，早已超越了国界。范围广泛的、复杂的分布式应用正在Web环境中出现。Web的流行和无所不在，是因为它能提供支持所有类型内容连接的信息发布，容易为最终用户存取。

Yogesh Deshpande和Steve Hansen在1998年就提出了Web工程的概念。Web工程作为一门新兴的学科，提倡使用一个过程和系统的方法来开发高质量的基于Web的系统。它"使用合理的、科学的工程和管理原则，用严密的和系统的方法来开发、发布和维护基于Web的系统"。目前，对于Web工程的研究主要是在国外开展的，国内才刚刚起步。

在基于Web的系统开发中，如果缺乏严格的过程，我们在开发、发布、实施和维护Web的过程中，可能就会碰到一些严重的问题，失败的可能性很大。而且，随着基于Web的系统变得越来越复杂，一个项目的失败将可能引发很多问题。当这种情况发生时，我们对Web和Internet的信心可能会动摇，从而引起Web危机。并且，Web危机可能会比软件开发人员所面对的软件危机更加严重、更加广泛。

这里我们谈到的Web系统是指以Browser/Server的访问方式为主，包含客户端浏览器、Web应用服务器、数据库服务器的软件系统。首先从技术实现上来讲，一般的B/S结构，无论是.NET还是J2EE，都是多层构架，有界面层、业务逻辑层、数据层。其次，从结构上来讲，都有客户端部分、传输网络部分和服务器端部分。

一个典型的Web系统的结构示意图如图9-1所示。

- 访问客户端：包含用户操作的浏览器及运行平台。最常见的一个例子就是Windows XP+IE6.0，另外，还有Windows及其他平台上的Netscape、Opera、Mozilla等浏览器。
- Web应用服务器：用于发布Web页面，接受来自客户端的请求，并把请求的处

理结果返回客户端。一般采用的 Web 应用服务程序有各种版本 UNIX 上的 Apache、WebLogic，Windows 服务器上的 Tomcat、IIS 等。

- 数据库：虽然数据库不是 Web 系统一个必要的部分，但在现有的大多数 Web 系统中，数据库是一个重要的部分。数据库多为关系型数据库，常用的有 Oracle、SQL Server、Sybase、Informix 等。
- 网络及中间件：提供客户端的请求到 Web 服务器的通道。网络可以是 Internet/Intranet/Extranet 网，也可能是局域网。中间件常常是传输中间件或交易中间件。
- 防火墙与 CA 认证：系统的安全性的一个保障系统，对于重要的系统是必不可少的。

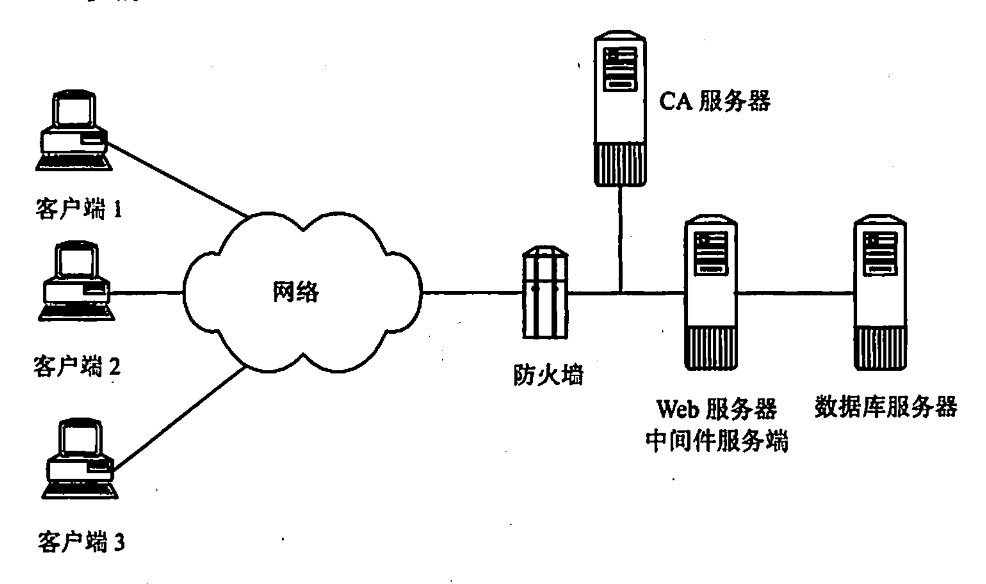


图 9-1　典型的 Web 系统

另外，一些大型 Web 系统，为了承受较大的访问压力，会采用负载均衡技术，使用多个 Web 应用服务器，分担来自客户端的访问压力。

9.1.2 Web 系统设计技术

当你在互联网上冲浪的时候，细心观察一下，不难发现网页的制作技术在不断进步，日新月异。从以往单调的 HTML，到 Java 小程序、Javascript 脚本、ActiveX 控件的加入，使本来静态的网页变得“动”了起来。滚动字幕、日历、划动鼠标时出现的轨迹等，还有近来许多网页上所使用的 FLASH 动画，令我们的网络世界越来越精彩。



现在我们来全面体验一下关于网页制作技术的服务端编程开发，也就是 Web 应用程序的设计。网页计数器、问卷调查系统、BBS 论坛、聊天室虚拟社区，相信大家都并不陌生。Web 应用程序为网页提供了真正的交互能力。不难想象，如果没有了这些给我们提供极大交互性的 Web 程序，那么浏览网页也就会变得不再那么有趣了，而对于网站的商业价值也就似乎只能提供像报纸媒体一样的宣传效果，不能互动，不能在线收集用户意见，更别说实现网上的电子商务，在线购物，销售产品了。

1．静态页面与动态页面

在网站建设发展的初期，人们全部使用 HTML 语言设计网页。这些简单的 HTML 文档被手工编辑完成之后，保存在与 Internet 相连接的计算机内，任何连入互联网的用户都可以访问其中的信息，这种形式的页面被称作静态页面。

现在，我们不仅需要 Web 提供所需的信息，还需要提供例如个性化搜索、收发 email、进行网上销售、从事电子商务等功能。为实现以上功能，必须使用更新的网络编程技术制作动态网页。所谓动态，指的是按照访问者的不同需要，对访问者输入的信息作出不同的响应的信息。

2．网络开发技术

网络开发技术一般指采用脚本语言进行编程的技术。脚本语言具有和传统的编程语言相似的语法结构和风格，并且可以结合 HTML 语言共同使用。脚本语言与 HTML 语言具有非常好的兼容性，使用者可以直接在脚本代码中加入 HTML 标签，或者在 HTML 标签中加入脚本代码从而更好地实现页面控制。

使用不同技术编写的动态页面也被保存在 Web 服务器内。当客户端用户向 Web 服务器发出访问动态页面的请求时，Web 服务器将根据用户所访问页面的后缀名确定该页面所使用的网络编程技术，然后把该页面提交给相应的解释引擎；解释引擎扫描整个页面找到特定的定界符，并执行位于定界符内的脚本代码以实现不同的功能，如访问数据库，发送电子邮件，执行算术或逻辑运算等，最后把执行结果返回 Web 服务器；最终，Web 服务器把解释引擎的执行结果连同页面上的 HTML 内容以及各种客户端脚本一同传送到客户端。

ASP（Active Server Pages，即动态服务器页面）是由微软公司推出的一种网络编程技术。准确地说，ASP 不能算作是一种编程语言，因为 ASP 没有提供自己专门的编程语言，而是允许用户使用包括 VBScript，JavaScript 等在内的许多已有的脚本语言编写 ASP 的应用程序。因此，ASP 实际上应当是一种脚本语言的服务端编写环境。ASP 吸收了当今许多流行的技术，如 IIS，Activex，VBScript，ODBC 等，是一种发展较为成熟的网络应用程序开发技术。ASP 的核心技术是对组件和对象技术的充分支持。通过使用 ASP 的组件和对象技术，用户可以直接使用 ActiveX 控件，调用对象方法和属性，以简单的

方式实现强大的功能。

JSP（Java Server Pages）是由Sun Microsystem公司于1999年6月推出的新技术，是基于Java Servlet以及整个Java体系的Web开发技术。ASP一般只应用于Windows NT/2000平台，而JSP则可以不加修改地在85%以上的Web Server上运行，其中包括了NT的系统，符合"一次编写，多平台运行"的Java标准，实现平台和服务器的独立性，而且基于JSP技术的应用程序，比基于ASP的应用程序易于维护和管理。

ColdFusion是Allaire公司在1995年推出的产品。它包含一个集成的可视化开发环境，可以极大地简化用户的开发过程。此外，ColdFusion采用一种被称做ColdFusion标识语言（ColdFusion Markup Language，CFML）的技术。CFML技术继承了HTML语言的风格，用户完全可以像使用HTML标签一样，使用CFML标签来完成各种复杂的操作。可以说，只要掌握了各种CFML标签，也就等于掌握了ColdFusion技术的精髓。

文字分析报告语言（Practical Extraction and Report Language，PERL），是现在编写CGI程序最受欢迎的编程工具，它具有强大的字符串处理能力，特别适合用于分割处理客户端Form提交的数据串，而且可以与ColdFusion等多种流行的Web数据库处理软件相结合。使用PERL既可以达到跨服务平台，又能方便地处理和使用数据库，还有一点，PERL是一种由解释器直接解释执行的语言，不同于使用C或VB等需要预编译后才能执行。

超文本预处理器（Hypertext Preprocessor，PHP）在很大程度上综合了PERL、Java和C语言的精华，在语法架构上继承了C语言的风格。PHP是一种自由软件。

3．CGI（通用网关接口）程序

CGI是一种早期用于Web程序设计的接口标准，也就是说，你只要熟悉这套接口标准，就能使用任何平常你所熟悉的编程语言来编制CGI程序。比如说C、VB、DELPHI等，当然，要使用CGI，也需要相应的Web服务器支持这一标准，常用的Web服务器软件，如著名的Apache，Web-site，Netscape Enterprise Server以及Microsoft的IIS等，都能很好地支持各种CGI程序。

CGI程序与Web服务器的交互主要有两种数据交换方式。

在UNIX或Linux下，是通过标准输入/输出来实现的，因此可以在程序中直接通过标准输入来取得客户端传来的请求和所传递的数据，然后在程序中对数据进行处理。比如说，你需要编制一个留言部的CGI程序，不外乎就是从客户端提交的Form表单中取得用户的名字、电子邮件，以及留言内容，而服务端相应的CGI程序则从标准输入中取得用户所提交的内容，而通过服务器的环境变量，你可以获得大量的客户端资料，如客户端的IP地址、计算机名字、所使用的浏览器，如果需要使用服务器认证功能，环境变量还能为你提供客户端用户的登录名字以及口令等。取得用户提交内容后，进行记录留

言的操作，打开数据文件，把留言内容写入文件，然后通过标准输出把“留言成功”的HTML信息输出到客户端。

而在Windows平台下，CGI程序与Web服务器的数据交互则是通过profile文件来进行的，profile的格式如同win.ini的格式一样。当Web服务器接收到客户端的请求数据后，就把它们以key-value的INI文件格式保存至暂存文件以供CGI程序来接收。因此，开发Windows CGI程序的第一步就是对数据文件进行拆解，如果是使用PERL的话，那么这一步可以免除了，因为PerlFor Windows的解释器已为你做好了这一步，你只需要如同在UNIX下一样，把数据作为标准输入来处理就可以了，也就是说，只要在PERL解释器所支持的功能范围内，UNIX下的PERL程序基本上可以不经修改地作为Windows CGI程序移植到Windows平台运行，这是PERL非常受CGI程序员喜爱的原因之一。

4．J2EE

J2EE的全称是Java2EnterpriseEdition，它是由SUN公司领导，各厂商共同制定，并得到广泛认可的工业标准。业内许多大的应用服务器厂商如IBM、BEA、Oracle等都积极地参与J2EE标准的制定和实施工作。

J2EE是专门为企业应用制定的标准，企业可以用它来编制企业级的应用，它为企业应用提供了数据库存取，交易完整性，可靠消息传递等功能。从公元2000年开始，越来越多的企业把自己的关键应用开始构建在支持J2EE标准的Web应用服务器之上。现在主流的一些J2EE应用服务器都可以支持企业应用所需的稳定性、可用性、安全性、可靠性、可扩展性等。

5．.NET

.NET的前身是微软的DNA（Distributed Network Architecture），在2000年的时候被.NET所取代。

.NET以WebServices为核心，全面支持SOAP、UDDI和WSDL，并在底层实现了类似Java虚拟机的CLR（Common Language Runtime）和一套具有3500多个类的.NET基础类库，以支持其开发语言Visual Basic、C#和ASP等，并且在后端支持其数据库SQL Server和Passport，.NET Studio组成了完整的解决方案。

6．WebServices

Web Services是建立可互操作的分布式应用程序的新平台。Web Services是一场分布式计算模式的跃进，它真正要解决分布式计算的问题。Web Services技术成为企业与企业之间连接的桥梁，为跨企业、跨行业、跨地域的业务提供了技术的实现方法。

Web Services采用XML作为消息交换的格式，利用Internet上的通用传输协议（如HTTP），提供标准的服务接口调用的方法，为客户提供灵活、方便、强大的Web服务。

9.1.3　Web 系统的测试策略

针对 Web 系统的构成和 Web 系统的一些特点，我们需要对涉及 Web 系统质量的各个方面进行测试。按系统架构来分，可分为客户端的测试、服务器端的测试和网络上的测试；按职能来分，可分为应用功能的测试、Web 应用服务的测试、安全系统的测试、数据库服务的测试；按软件的质量特性来分，又可分为功能测试、性能测试、安全性测试、兼容性测试和易用性测试；按照开发阶段来分，可以分为设计的测试、编码的测试和系统的测试。

Web 是一类特殊的软件，Web 应用系统的开发同样要经过需求分析、设计、编码、实施等阶段，所以对软件的测试是贯穿整个软件生命周期的。Web 应用系统的设计、编码和实施三个阶段都是十分重要，现把对 Web 应用系统的测试分为 Web 应用设计测试、Web 应用开发测试和 Web 应用运行测试，在后面的章节中将分别进行论述。

9.2　Web 应用设计测试

9.2.1　Web 应用设计测试概述

前面已经谈到，即使最简单的一个 Web 应用系统也会包括客户端、中间件、服务器等几部分，如果要设计一个大型的复杂应用，则可能包含数据库服务器、应用服务器、认证服务器等多个组成部分。在 Web 应用系统的设计阶段，测试的一个主要内容就是对 Web 设计从全面性、适合性、标准性等多方面进行检查。依据 Web 应用系统的架构，把对 Web 设计的测试分为总体架构设计的测试、客户端设计的测试、服务器端设计的测试三个部分。

9.2.2　总体架构设计的测试

Web 应用系统中 PC 模型的复杂性呈指数增加，除了面临多个客户 PC 机所带来的测试挑战外，Web 系统的服务端还涉及各种类型的硬件及操作系统、服务进程、服务器包和数据库等的软件组合。对总体设计的检查从以下几个方面进行。

1．采用瘦客户端或胖客户端是否适合需求

瘦客户端与胖客户端是指部分应用程序和组件是否驻留在客户端。在瘦客户端系统中，客户端 PC 只作少量的处理，业务逻辑规则多数在服务器端执行。多数 Web 新闻站点、门户网站及用于信息发布的 Web 系统采用这种模式。这种模式适合于对客户端没有特殊要求、用户量庞大并且分散的 Web 应用系统。

胖客户端既运行应用程序的用户界面，又执行部分业务逻辑。此时，浏览器不仅要处理 HTML 等页面，还要执行 Java applet 和 ActiveX 控件等其他组件。一些银行客户系统、网络游戏、网上办公系统等 Web 应用系统采用这种模式。这种模式适合于对安全性要求较高、交互操作频繁或业务逻辑复杂的 Web 应用系统。胖客户端能减轻服务器端的处理强度，与服务器端之间有较强的交互能力，但需要在客户端安装一些应用程序或组件。

确定采用哪种模式，要由 Web 应用系统的具体需求决定。测试的任务就是验证设计中采用的模式是否符合系统需求。

2．确定 Web 架构的组成部分是否满足需求

根据业务需求，服务器端可能是一台简单的 Web 应用服务器，也可能是由中间件、数据库和安全服务器等组成的服务器群。在确定服务器端的组成部分时，需要考虑到成本、功能、安全性要求、容量要求、传输实时性等多个方面。对 Web 架构设计的测试除了要验证 Web 架构的组成部分是否满足上述需求外，还要检查各组成部分是否有搭配不兼容的地方。

3．服务器的配置及分布是否满足需求

服务器软件可能分布在若干个物理服务器单元上。如表 9-1 所示列出了服务器配置的几个例子。

表 9-1 Web 服务器端的配置及分布

序号	模　型	单元 1	单元 2	单元 3
1	单单元模型	基于 NT 的 Web 服务器 基于 NT 的应用服务器 基于 NT 的数据库服务器		
2	双单元模型	基于 NT 的 Web 服务器 基于 NT 的应用服务器	基于 NT 的数据库服务器	
3	三单元模型	基于 NT 的 Web 服务器 基于 NT 的应用服务器	基于 NT 的 Web 服务器 基于 NT 的应用服务器	基于 NT 的数据库服务器
4	单单元模型	基于 UNIX 的 Web 服务器 基于 UNIX 的应用服务器 基于 UNIX 的数据库服务器		
5	双单元模型	基于 UNIX 的 Web 服务器 基于 UNIX 的应用服务器	基于 UNIX 的数据库服务器	
6	三单元模型	基于 NT 的 Web 服务器 基于 NT 的应用服务器	基于 NT 的 Web 服务器 基于 NT 的应用服务器	基于 UNIX 的数据库服务器

这部分测试的重点是验证服务器端的配置和分布是否满足用户的功能、性能、成本

等需求。

9.2.3　客户端设计的测试

客户端设计主要是面向用户的，包括功能设置、信息组织结构设计和页面设计。对客户端设计的测试也将从这三方面进行。

1．功能设置的测试

Web 应用系统的功能设置应尽量以企业内部工作的需求为主，取消或少设与工作无关的功能（如网上游戏、网上聊天等），以提高工作效率。常用的功能有以下几类。

- 信息服务。

此类功能为读者提供多种动、静态信息，如公共信息、同行业信息、企业内部管理信息、业务信息、信息搜索等。

- 办公自动化。

此类功能主要提供企业内部通信、工作流控制等方面的功能，以实现企业办公自动化（实现无纸办公），如电子邮件、公文管理、内部论坛、会议日程安排、任务追踪、综合审批流程等。

- Internet 支持。

企业内部网通过设立防火墙、代理服务器等接入 Internet 网，使员工享受 Internet 上提供的各种服务，扩大信息获取渠道，弥补 Intranet 站点上功能及信息的不足。

这部分的测试重点是验证功能设置能否满足用户需求。

2．信息组织结构设计的测试

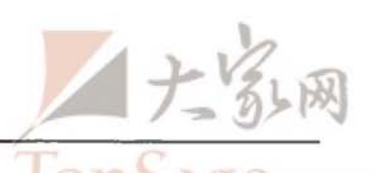
信息是 Web 应用系统，特别是 Web 站点的灵魂，信息发布是 Web 应用系统的一个重要功能。在进行 Web 应用系统设计时，要确保把需要发布的信息按功能、主题等分类，并以某种方式结构化相应的信息，使用户能直观、快捷地查到所需信息。信息组织结构设计主要有线性结构设计、分层结构设计和非线性结构设计三种。

线性结构信息的组织方式与一本书的组织方式相似，信息按顺序链接，一页紧接另一页。它限制用户不能以非顺序（即非线性）方式浏览内容，但可允许用户向前或向后搜索信息内容。

分层结构信息以树形方式组织信息，采用线性路径搜索方式浏览信息内容。它具有层次清楚、易于查找等特点，搜索路径只能向上或向下。

非线性结构信息按内容的关系链接信息，没有明显的结构。它允许用户随意地在信息中漫游。用户可在非线性结构中完全自由地改变信息浏览路径。

对信息组织结构设计的测试重点是验证设计模式是否适合信息的特点，能否使用户直观、快捷地浏览到所需信息。



3. 页面设计的测试

页面是信息的载体，它直接体现 Web 站点的设计水平。一个好的页面因信息层次清晰而让用户一目了然；因设计巧妙、精致美观而让用户流连忘返；因恰当使用各种元素能完成许多功能而不显拥挤。对页面设计的测试可从以下几个方面进行：

- 页面的一致性如何；
- 在每个页面上是否设计友好的用户界面和直观的导航系统；
- 是否考虑多种浏览器的需要；
- 是否建立了页面文件的命名体系；
- 是否充分考虑了合适的页面布局技术，如层叠样式表、表格和帧结构等。

9.2.4 服务器端设计的测试

服务器端的设计包括众多的内容，而且都是非常重要的内容，例如数据库设计、安全系统设计等，这些设计会在很大程度上影响到 Web 应用系统的性能、安全性、可靠性等，因此要对这些设计进行严格的测试。下面分别讲述对容量规划、完全系统设计和数据库设计的测试。

1. 容量规划的测试

大多数进程被划分为两类：输入输出限制（I/O-bound）和 CPU 限制（CPU-bound）。用于静态 HTML 的常常是 I/O-bound，从硬盘检索文件的速度受到该速率的限制，同时文件从网络接口移出时受到该速度的限制。动态 HTML 的产生恰好相反，它常常是 CPU-bound 的，也就是说它用于产生页面的时间要比将页面移出网络接口的时间要长。CPU 此时的作用很关键，特别是用 CGI 或 Java Servlet 来创建动态页面时尤其明显，并且大多数此类 CPU 处理是串操作。另一方面，依赖于数据库查询的动态内容常常受到数据库速度的限制，数据库又通常是 I/O-bound 的，因为它需要从硬盘上检索数据。容量规划就是把用户的需求量化成具体指标，作为 Web 设计的一个重要依据。

进行容量规划是非常必要和重要的，它和 Web 应用系统的性能是息息相关的。对容量规划的测试也非常必要，评价容量规划设计的关键在于：将所要求的延迟和带宽与该体系结构中每一环节的额定容量作一比较，每个组成部分都必须满足这些要求，其中还必须要考虑到系统各部分之间可能产生的内耗，以及对该体系的生命周期可能增加的负载。对容量规划的测试方法是，检查容量规划是否满足用户的需求，可以从以下几个方面进行检查。

- 估算点击率是否满足需求。

点击率就是每秒 HTTP 的请求数，也叫每秒被访问的次数。从统计学的角度来看，服务器的负载是时间的函数，根据 Web 内容和用户群的分布，时间函数表现的曲线是不

同的。对点击率的估算可采用80～20原理和UCML方法（参见本书负载压力测试部分内容），根据用户的需求估算容量规划中的点击率是测试的一个主要方法。

- 估算延迟和流量是否满足需求。

规划流量及延迟目标必须要考虑用户访问网络的能力及期望值，因此这部分测试就是对用户的情况进行分析，检查容量规划中确定的流量及延迟是否能达到用户的期望值。如果容量规划是以宽带接入的用户的标准进行设计的，那么这些指标对使用调制解调器上网的用户来说就不适用了，可能延迟时间会超过他们的接受范围。

在估算时还要注意Web应用系统设计中是否提供流媒体，流媒体需要连续传输，可能持续几分钟的时间，这和标准HTTP传输的连接时间有很大不同。另外，流媒体有不同的格式，例如音频和视频，它们的估算公式和标准也是不一样的。它们会占用大量带宽，对延迟时间有严格的要求。

- 估算Web应用系统所需服务器的资源消耗。

服务器的内存是需要重点考虑的，经常是性能瓶颈所在。当内存不足时，有的进程会转移到硬盘上去运行，造成性能急剧下降。而且，一个缺少内存的系统常常表现出很高的CPU利用率，因为它需要不断地扫描内存，将内存中的页面移到磁盘上。估算所需的内存需要考虑几个方面：操作系统所需内存、Web 高速缓存所需内存、CGI 所需内存等。

2．安全系统设计的测试

安全是贯穿Web应用系统设计、运行、管理等全过程的一个非常重要的因素，需要从多方面考虑，才能制定出一个较为完善的安全策略，保证Web应用安全运行。因此，在对Web应用系统设计的测试中，对安全系统设计的测试是一个重要内容，一般从以下几个方面进行审核和评估。

① 常识性安全策略。评估设计中对重要的部分是否采取了取消不必要的协议、严格控制写权限、取消服务器目录浏览属性、保留日志记录等安全措施。

② 使用加密技术。为保证重要数据在网上传输过程中不被人窃听或修改，必须对数据进行加密，确保数据传输的安全性。加密的方法很多，如公共密钥加密、数字签名、链加密、文档加密、SSL和SHTTP等。审核采用的加密方式能否满足用户需求。

③ 构造防火墙。当Web应用系统与外部网络连接时，还必须构造防火墙，一般防火墙保护分为网络级、应用级和电路级等三种防方式。审核采用的防火墙方式和安全级别能否满足用户需求。

④ 构建网络防毒体系。病毒在网络中存储、传播、感染的途径多、速度快、方式各异，对Web系统的危害较大。因此，需要构建必要的网络防毒体系。验证网络防毒体系设计是否防护全面、多层次和全方位。



3．数据库设计的测试

数据库设计是一个重要内容，对数据库设计的测试在本书的相关章节论述，这里不再赘述。

9.3 Web 应用开发测试

9.3.1 Web 应用开发测试概述

Web 应用开发测试主要指在 Web 应用的开发阶段，对 Web 应用的源代码和组件进行测试，保证代码的正确性，组件的功能正常。在后面我们给出一个利用 Junit 进行 Java 开发的代码进行单元测试的例子。

9.3.2 代码测试

对 Web 应用系统的代码测试主要包括以下几个方面：源代码规则分析、链接测试、页面静态对象测试等，下面分别叙述。

1．源代码规则分析

主要方法是使用基于规则检查的工具，读取输入的源代码，然后将源代码与编码标准或语言规则相对照，以找出存在于两者之间的不一致性，或者存在于源代码当中的潜在错误。从某种意义上说，这类工具与文字处理器中的语法和拼写检查器十分相似。

HTML 校验工具通常与下面将要使用的链接验证工具集成在一起，主要有"Watchfire Linkbot"、"ParaSoft SiteRuler"、"Matterform Media Theseus" 等软件。另外，互联网上还有很多免费在线验证 HTML 和检查链接的服务网站，如 http://validator.w3.org、http://watson.addy.com 等。

2．链接测试

链接是 Web 应用系统的一个主要特征，它是在页面之间切换和指导用户去一些不知道地址的页面的主要手段。链接测试可分为三个方面。首先，测试所有链接是否按指示的那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在；最后，保证 Web 应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面，只有知道正确的 URL 地址才能访问。

测试方法很简单，就是逐一检查链接的有效性、可达性、正确性等，链接测试可以自动进行，有许多可代替手工操作的链接验证工具，在功能测试部分有所介绍。

3．框架测试

主要检查以下几个方面：

- 是否可随浏览窗口的变化自动调整大小；
- 在当前窗口不能完全显示内容时是否提供滚动条功能；
- 能否在正确的目标框架中打开新页面。

4．表格测试

主要检查表格能否随浏览器窗口的变化或页面的变化自动调整大小。

5．图形测试（Graphics）

主要检查点如下：

- 颜色饱和度和对比度是否合适；
- 需要突出的链接的颜色是否容易识别；
- 是否正确加载所有的图形。

9.3.3　组件测试

由于“组件”的概念被广泛使用，对于 Web 组件有很多歧义，我们有必要首先明确一下 Web 组件（Web component）在本书中的定义。

所谓 Web 组件是指这样一个软件单元：它被用于 Web 系统中，通常嵌入页面中，有些组件为完成一个特定的功能而存在于 Web 页面中或服务器上，用户的使用请求可以通过浏览器的解释传递给组件，组件执行的结果经浏览器传递给用户。

Web 组件范围是非常广的，随着 Web 系统设计技术的更新在不断发展着。例如我们常见的 Java applet、ActiveX 控件、VB 脚本、Javascript、各种插件、外接件等都属于此列。随着 Web 编程技术的更新，越来越多的第三方程序推出了支持 Web 的组件，例如 RealPlayer 插件、QuickTime 插件等。

对于一般的组件测试来说，可分为外形测试和交互测试，也可称为静态测试和动态测试。由于 Web 组件的范围十分广泛，相互之间差异较大，其测试方法也因对象不同而有所区别，我们将在后面重点介绍几种组件的测试方法。

1．表单（Forms）测试

当用户给 Web 应用系统管理员提交信息时，就需要使用表单操作，例如用户注册、登录、信息提交等。在这种情况下，我们必须测试提交操作的完整性，以校验提交给服务器的信息的正确性。例如：用户填写的出生日期与职业是否恰当，填写的所属省份与所在城市是否匹配等。如果使用了默认值，还要检验默认值的正确性。如果表单只能接受指定的某些值，则也要进行测试。例如，只能接受某些字符，测试时可以跳过这些字符，看系统是否会报错。

除了测试表单实现的功能，还要检查 Form 区域的外部表现，如文字环绕、随窗口的大小调整大小等。



2．Cookies 测试

Cookies 通常用来存储用户信息和用户在某应用系统的操作，当一个用户使用 Cookies 访问了某一个应用系统时，Web 服务器将发送关于用户的信息，把该信息以 Cookies 的形式存储在客户端计算机上，这可用来创建动态和自定义页面或者存储登录等信息。

如果 Web 应用系统使用了 Cookies，就必须检查 Cookies 是否能正常工作。测试的内容可包括 Cookies 是否起作用，是否按预定的时间进行保存，刷新对 Cookies 有什么影响等。

3．脚本测试

由于脚本采用不同的语言编写，脚本的测试可以分为静态测试和动态测试。

静态测试是指手工或利用工具检查脚本的源代码的语法错误、逻辑错误和其他与语言有关的编程错误，通常在单元测试阶段实施，由开发人员执行。有很多基于其他编程和脚本语言的，基于规则的分析器可用来进行静态测试，如检查 VB 脚本的“Compuware Numega CodeReview”、检查 C/C++脚本的“ParaSoft Codewizard”、检查 Java 的 “ParaSoft Jtest” 等。

动态测试指对 Java、Javascript、 VBScript 、C/C++或 Perl 等脚本的功能进行逐一验证。需要特别注意的是，不同的浏览器可能要加一定的插件或补丁才能支持上述的一些脚本。

4．CGI 测试

CGI 是一种服务器端技术，给予用户一定的交互操作权利。实际上 CGI 是一种协议，用户可以通过浏览器去实施在服务器上的一些操作，包括运行服务器上的.exe 程序。所以测试 CGI 时，可以在服务器上使用一些监控器查看执行 CGI 的结果。

同样，CGI 也会遇到性能问题，在需要的时候也要对 CGI 进行性能测试。

5．ASP 测试

ASP 实际上是一种含有脚本命令的文本文件，以.asp 作为文件扩展名，它缺省的输出是 HTML 或 HTML 及可在客户端处理的脚本。

测试 ASP 时，可先作代码检查，发现语法及其他明显的编码错误，然后再用不同的浏览器进行验证，以发现 ASP 是否工作正常。在测试 ASP 时需要注意以下几个方面。

- 注意浏览器的缓存问题。缓存设置有时会影响到某些 ASP 的执行。
- 正确设置超时。超时会导致一些 ASP 页面失效，从而不能正确执行。
- 一些 ASP 的性能测试是非常必要的。如可能出现非常频繁的数据库查询等。ASP 的性能测试可放到 Web 应用系统的性能测试中考虑。

6．ActiveX 控件测试

ActiveX 控件是客户端技术，实际上类似于 Win32 程序中的 OCX 控件，也是一种很常用的控件，在很多 Web 系统设计中都会用到。我们在测试 ActiveX 控件时要注意以下几个方面。

- ActiveX 只能用于 Windows 客户端，如果我们使用诸如 Netscape 的浏览器时，需要添加相应的插件。
- ActiveX 应用是编译后的应用，在用户浏览时需要下载到客户端运行，因此，客户端的安全设置可能会影响到 ActiveX 控件的使用。测试时需要保证 ActiveX 控件的签名注册通过验证。
- 由于用户有权拒绝使用 ActiveX 控件，所以，需要测试在 ActiveX 控件没有下载时，Web 系统的功能受影响的程度。
- 要专门进行 ActiveX 控件的安装与卸载测试，考察安装与卸载过程是否能顺利进行。
- 测试之前需要确认测试环境中没有旧版本的控件，如果有，一定要卸载后进行重新安装测试。因为 ActiveX 控件与其他组件不同，在网页更新时不会自动重新安装。

9.3.4　使用 Junit 进行单元测试

越来越多的 Web 应用采用 Java 技术开发，下面以 Junit 为例讨论 Java 开发的 Web 应用的单元测试。

Junit Framework 是一个被多数 Java 程序员采用和证实的优秀的测试框架，可以用 Junit Framework 来编写适应自己开发项目的单元测试程序。Junit 的用户指南，重点在于解释单元测试框架的设计方法以及简单的类使用说明，而对在特定的测试框架下如何实施单元测试，如何在项目开发的过程中更新和维护已经存在的单元测试代码没有详细的解释，下面就这两个需要解决的问题进一步说明。

1．单元测试的编写原则

Junit 附带文档所列举的单元测试带有一定的迷惑性，因为几乎所有的示例单元都是针对某个对象的某个方法，似乎 Junit 的单元测试仅适用于类组织结构的静态约束，从而使初学者怀疑 Junit 下的单元测试所能带来的效果。因此，我们需要重新定义如何确定有价值的单元测试，以及如何编写这些单元测试、维护这些单元测试，从而让更多的程序员接受并熟悉 Junit 下的单元测试的编写。

在进行 Junit 单元测试框架的设计时，设定了如下三个总体目标：

① 简化测试框架的编写，这种简化包括测试框架的学习和实际测试单元的编写；

② 使测试单元保持持久性；

③ 可以利用已有的框架来编写相关的框架。

从这三个目标可以看出，单元测试框架的基本设计考虑依然是从我们现有的测试方式和方法出发，而只是使测试变得更加容易实施、扩展并保持持久性。因此编写单元测试的原则可以借鉴和利用我们通常使用的测试方法。

2. 如何确定单元测试

在通常的测试中，一个单元测试一般针对于特定对象的一个特定特性，例如，假定编写了一个针对特定数据库访问的连接池的类包实现，则要建立以下的单元测试。

- 在连接池启动后，是否根据定义的规则在池中建立了相应数量的数据库连接；
- 申请一个数据库连接，是否根据定义的规则从池中直接获得缓存连接的引用，还是建立新的连接；
- 释放一个数据库连接后，连接是否根据定义的规则被释放以便以后使用；
- 后台 Housekeeping 线程是否按照定义的规则释放已经过期的连接申请；
- 如果连接有时间期限，后台 Housekeeping 线程是否定期释放已经过期的缓存连接。

这里只列出了部分的可能测试，但是可以看出单元测试的粒度。一个单元测试基本是以一个对象的明确特性为基础，单元测试的过程应该限定在一个明确的线程范围内。根据上面所述，一个单元测试的测试过程非常类似于一个 Use Case 的定义，但是单元测试的粒度一般来说比 Use Case 的定义要小，这点是容易理解的。因为 Use Case 是以单独的事务单元为基础的，而单元测试是以一组聚合性很强的对象的特定特征为基础的。一般而言，一个事务中会利用许多的系统特征来完成具体的软件需求。

从上面的分析可以得出，测试单元应该把一个对象的内部状态的转换为基本编写单元。一个软件系统就和一辆设计好的汽车一样，系统的状态是由同一时刻系统内部的各个分立的部件的状态决定的，因此为了确定一个系统最终的行为符合要求，我们首先需要保证系统内的各个部分的状态会符合我们的设计要求，所以我们的测试单元的重点应该放在确定对象的状态变换上。

然而，需要注意的并不是所有的对象组特征都需要被编写成独立的测试单元，应该在有可能引入错误的地方引入测试单元，通常这些地方存在于有特定边界条件、复杂算法以及需求变动比较频繁的代码逻辑中。除了这些特性需要被编写成独立的测试单元外，还有一些边界条件比较复杂的对象方法也应该被编写成独立的测试单元，这部分单元测试已经在 Junit 文档中被较好地描述和解释过了。

在基本确定了需要编写的单元测试之后，我们还应该问自己：编写好了这些测试，我们是否可以有把握地告诉自己，如果代码通过了这些单元测试，我们能认定程序的运

行是正确的，符合需求的。如果我们不能非常地确定，就应该看看是否还有遗漏的需要编写的单元测试，或者重新审视我们对软件需求的理解。通常来说，在开始使用单元测试的时候，大多数单元测试总是没有错误的。

一旦我们确定了需要编写的测试单元，接下来就应该着手编写。

3．如何编写单元测试

通常强调单元测试必须由类包的编写者负责编写，这个限定对于我们设定的测试目标是必须的。因为只有这样，测试才能保证对象的运行行为符合需求，而通过类接口的测试，我们只能确保对象符合静态约束，因此这就要求我们在测试的过程中，必须开放一定的内部数据结构，或者针对特定的运行行为建立适当的数据记录，并把这些数据暴露给特定的测试单元。这也就是说我们在编写单元测试时必须对相应的类包进行修改，这样的修改也发生在我们以前使用的测试方法中，因此，以前的测试标记及其他一些测试技巧仍然可以在 Junit 测试中改进使用。

由于单元测试的总体目标是保证软件在运行过程中的正确无误，因此在我们对一个对象编写单元测试的时候，我们不但需要保证类的静态约束符合我们的设计意图，而且需要保证对象在特定条件下的运行状态符合我们的预先设定。下面还是拿数据库缓冲池的例子进行说明。一个缓冲池暴露给其他对象的是一组使用接口，其中包括对池的参数设定、池的初始化、池的销毁、从这个池里获得一个数据连接，以及释放连接到池中，对其他对象而言，随着各种条件的触发而引起池的内部状态的变化是不需要知道的，这一点也是符合封装原理的。但是池对象的状态变化，例如缓存的连接数在某些条件下会增长，一个连接在足够长的运行后需要被彻底释放，从而使池的连接被更新等，虽然外部对象不需要明确，但是却是程序运行正确的保证，所以我们的单元测试必须保证这些内部逻辑被正确地运行。

编译语言的测试和调试是很难对运行逻辑过程进行跟踪的，但是我们知道，无论逻辑怎么运行，如果状态的转换符合我们的行为设定，那么结果显然是正确的。因此，在对一个对象进行单元测试的时候，我们需要对多数的状态转换进行分析和对照，从而验证对象的行为。状态是通过一系列的状态数据来描述的，因此编写单元测试，首先分析出状态的变化过程（状态转换图对这个过程的描述非常清晰），然后根据状态的定义确定分析的状态数据，最后是提供这些内部的状态数据的访问。在数据库连接池的例子中，我们对池实现的对象“DefaultConnectionProxy”的状态变换进行分析后，我们决定把表征状态的“OracleConnectionCacheImpl”对象公开给测试类。参见示例 1。

【示例 1】

```
/**
 * 这个类简单地包装了 Oracle 对数据连接缓冲池的实现。
```




```
*
*/
public class DefaultConnectionProxy extends ConnectionProxy {
private static final String name = "Default Connection Proxy";
private static final String description =
"这个类简单地包装了 Oracle 对数据连接缓冲池的实现。";
private static final String author = "Ion-Global.com";
private static final int major_version = 0;
private static final int minor_version = 9;
private static final boolean pooled = true;
private ConnectionBroker connectionBroker = null;
private Properties props;
private Properties propDescriptions;
private Object initLock = new Object();
// Test Code Begin...
/*
```

为了能够了解对象的状态变化，需要给表征对象内部状态变化的部分私有变量提供公共的访问接口（或者提供让同一个类包访问的接口），以便使测试单元可以有效地判断对象的状态转变，在本示例中对包装的 OracleConnectionCacheImpl 对象提供访问接口。

```
*/
OracleConnectionCacheImpl getConnectionCache() {
if (connectionBroker == null) {
throw new IllegalStateException("You need start the server
first.");
}

return connectionBroker.getConnectionCache();
}
// Test Code End...
```

在公开内部状态数据后，我们就可以编写我们的测试单元了。单元测试的选择方法和选择尺度已经在前面章节进行了说明。但是仍然需要注意的是，由于 assert 方法会抛出一个 error，你应该在测试方法的最后集中用 assert 相关方法进行判断，这样可以确保资源得到释放。对数据库连接池的例子，我们可以建立测试类 DefaultConnection-ProxyTest，同时建立数个测试案例，如下所示。

【示例 2】

```
/**
* 这个类对示例 1 中的类进行简单的测试。
*
*/
public class DefaultConnectionProxyTest extends TestCase {
private DefaultConnectionProxy conProxy = null;
private OracleConnectionCacheImpl cacheImpl = null;
private Connection con = null;
/** 建立必要的测试起始环境。
*/
protected void setUp() {
conProxy = new DefaultConnectionProxy();
conProxy.start();
cacheImpl = conProxy.getConnectionCache();
}
/** 对示例 1 中的对象进行服务启动后的状态进行测试，检查在服务启动后，
连接池的参数设置是否正确。
*/
public void testConnectionProxyStart() {
int minConnections = 0;
int maxConnections = 0;
assertNotNull(cacheImpl);
try {
minConnections =
Integer.parseInt(PropertyManager.getProperty
("DefaultConnectionProxy.minConnections"));
maxConnections =
Integer.parseInt(PropertyManager.getProperty
("DefaultConnectionProxy.maxConnections"));
} catch (Exception e) {
// ignore the exception
}
assertEquals(cacheImpl.getMinLimit(), minConnections);
assertEquals(cacheImpl.getMaxLimit(), maxConnections);
assertEquals(cacheImpl.getCacheSize(), minConnections);
}
/** 对示例 1 中的对象进行获取数据库连接的测试，看看是否可以获取有效的数据库连接，并且
```



看看获取连接后，连接池的状态是否按照既定的策略进行变化。由于 assert 方法抛出的是 error 对象，因此，尽可能把 assert 方法放置到方法的最后，集体进行测试，这样在方法内打开的资源，才能有效地被正确关闭。

```
*/
public void testGetConnection() {
int cacheSize = cacheImpl.getCacheSize();
int activeSize = cacheImpl.getActiveSize();
int cacheSizeAfter = 0;
int activeSizeAfter = 0;
con = conProxy.getConnection();
if (con != null) {
activeSizeAfter = cacheImpl.getActiveSize();
cacheSizeAfter = cacheImpl.getCacheSize();
try {
con.close();
} catch (SQLException e) {
}
} else {
assertNotNull(con);
}
```

/*如果连接池中的实际使用连接数小于缓存连接数，检查获取的新的数据连接是否从缓存中获取，反之连接池是否建立新的连接。

```
*/
if (cacheSize > activeSize) {
assertEquals(activeSize + 1, activeSizeAfter);
assertEquals(cacheSize, cacheSizeAfter);
} else {
assertEquals(activeSize + 1, cacheSizeAfter);
}
}
```

/** 对示例 1 中的对象进行数据库连接释放的测试，看看连接释放后，连接池的状态是否按照既定的策略进行变化。由于 assert 方法抛出的是 error 对象，因此尽可能把 assert 方法放置到方法的最后集体进行测试，这样在方法内打开的资源，才能有效地被正确关闭。

```
*/
public void testConnectionClose() {
int minConnections = cacheImpl.getMinLimit();
int cacheSize = 0;
int activeSize = 0;
int cacheSizeAfter = 0;
```

```
int activeSizeAfter = 0;
con = conProxy.getConnection();
if (con != null) {
cacheSize = cacheImpl.getCacheSize();
activeSize = cacheImpl.getActiveSize();
try {
con.close();
} catch (SQLException e) {
}
activeSizeAfter = cacheImpl.getActiveSize();
cacheSizeAfter = cacheImpl.getCacheSize();
} else {
assertNotNull(con);
}
assertEquals(activeSize, activeSizeAfter + 1);
/*如果连接池中的缓存连接数大于最少缓存连接数，检查释放数据连接后，是否缓存连接数比之前减少了一个，反之缓存连接数是否保持为最少缓存连接数。
*/
if (cacheSize > minConnections) {
assertEquals(cacheSize, cacheSizeAfter + 1);
} else {
assertEquals(cacheSize, minConnections);
}
}
/** 释放建立测试起始环境时的资源。
*/
protected void tearDown() {
cacheImpl = null;
conProxy.destroy();
}
public DefaultConnectionProxyTest(String name) {
super(name);
}
/** 你可以简单地运行这个类从而对类中所包含的测试单元进行测试。
*/
public static void main(String args[]) {
junit.textui.TestRunner.run(DefaultConnectionProxyTest.class);
}
}
```



当单元测试完成后，我们可以用 Junit 提供的 TestSuite 对象对测试单元进行组织，你可以决定测试的顺序，然后运行你的测试。

4. 如何维护单元测试

通过上面的描述，我们对如何确定和编写测试有了基本的了解，但是需求总是变化的，因此我们的单元测试也会根据需求的变化不断地演变。如果我们决定修改类的行为规则，必然会对这个类的测试单元进行修改，以适应变化。但是，如果对这个类中仅有调用关系的类的行为定义没有变化，则相应的单元测试仍然是可靠和充分的，同时如果包含行为变化的类的对象的状态定义与其没有直接的关系，测试单元仍然起效。这种结果也是封装原则的优势体现。

9.4 Web 应用运行测试

9.4.1 Web 应用运行测试概述

Web 应用系统开发完成后，需要对 Web 应用进行全面的测试，其测试方法与其他系统的测试既有相同之处，又有不同之处。

相同之处体现在下面几个方面。

- 测试内容基本相同：Web 应用系统作为软件系统的一种形式，其测试内容也会包括功能、性能、易用性、兼容性和安全性测试等内容。
- 某些项目的测试方法基本相同：例如 Web 应用程序的功能测试与其他系统的功能测试方法是一样的，同样是根据功能说明书、需求说明书等文档，使用因果图法、边界值法等技术，设计测试用例进行测试。
- 测试手段基本相同：Web 应用系统的测试一样会采用人工测试、工具测试、评估等手段。

鉴于 Web 系统的自身特点，其测试与传统的软件测试也有所不同，使测试基于 Web 的系统变得困难。

首先是测试的重点不一样：Web 应用系统的性能可能是开发者或用户最关心的一个测试内容，由于 Internet 的不可预见性和用户连接数的不固定性，人们经常对 Web 系统的稳定水平有所担心。另外，一些不断发展中的 Web 设计技术也使 Web 组件测试变得重要。安全性对某些涉及交易或重要数据的 Web 应用系统也很重要。由于用户客户端的不确定性，易用性测试和客户端配置与兼容性测试也是必要的一个内容。

其次是测试采用的工具有所不同 Web 应用的一些测试，如链接测试、表单测试、界面测试等，通常采用可以重复执行的自动化工具进行，性能测试除了采用 LoadRunner、QALoad 等通用的负载压力测试工具外，还有很多专门用于 Web 系统性能测试的工具，

如 WAST（Web Application Stress Tool）、ACT（Application Center Test）、Webload 等。

最后 Web 应用系统迫切需要新的测试技术和方法：Web 应用系统的开发技术是更新最快的开发技术之一，针对这种新组件、新技术的测试手段也必须及时探索，甚至要开发出新的测试工具以满足需求。

9.4.2　功能测试

Web 应用功能测试指 Web 应用系统的基本功能的测试，其案例的设计方法可参见有关《黑盒测试技术》章节的内容。

考虑到 Web 应用本身的特点，其功能测试还要注意以下几个方面。

- 客户端的选择。

Web 应用客户端软件环境主要包括操作系统和浏览器。除非有特殊要求，测试功能时，我们一般选择比较流行的配置，如选择 WindowsXP+IE6.0 的简体中文版本。需要注意的是，浏览器的种类和版本有可能影响功能的正确实现。

- 客户端浏览器的配置。

一般情况下，开发者不会过多地考虑客户端配置问题，只是将更多的时间用于实现服务端的程序，而用户也往往不会刻意地对所使用的浏览器进行适应性配置。如果测试人员完全按照浏览器的缺省配置测试一个 Web 应用的功能，有可能会出现较多的因浏览器配置而引起的问题。下面以 IE6.0 为例进行说明，IE6.0 的主要配置界面如图 9-2 所示。

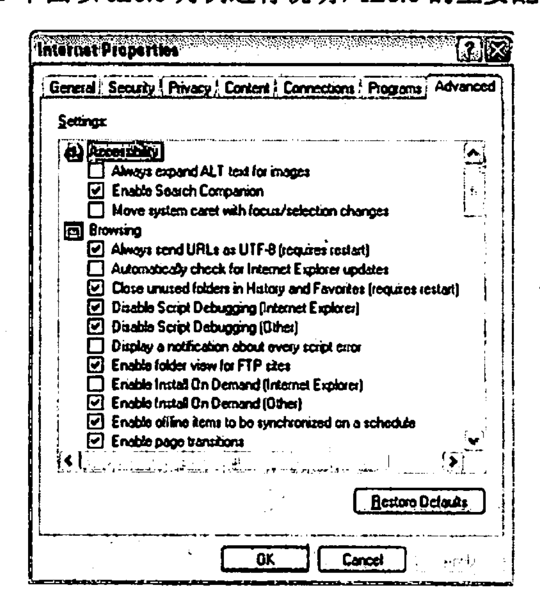


图 9-2　IE6.0 的主要配置界面



例如 Cookie 设置就会影响含有 Cookie 的 Web 应用的功能能否成功地实现。其他如脚本设置、安全设置、显示设置等大多数设置都会影响到 Web 应用功能的实现。

- 客户端的显示设置。

大多数人都喜欢使用 1024×768 像素的显示设置，但并不是所有的 Web 应用都支持这种设置。不合适的显示设置不但会使 Web 应用系统的界面显示异常，还可能导致应用功能无法实现。

- 内容测试。

由于 Web 应用带有一定的开放性，尤其是发布于互联网上的网站，其内容是完全开放的，因此在 Web 系统的功能测试中还要重点测试一个方面，即内容测试。

内容测试用来检验 Web 应用系统提供信息的正确性、准确性和相关性。信息的正确性是指信息是可靠的还是误传的。例如，在商品价格列表中，错误的价格可能引起财政问题，甚至导致法律纠纷；信息的准确性是指是否有语法或拼写错误。这种测试通常使用一些文字处理软件来进行，例如使用 Microsoft Word 的“拼音与语法检查”功能；信息的相关性是指，是否在当前页面可以找到与当前浏览信息相关的信息列表或入口，也就是一般 Web 站点中所谓的“相关文章列表”。

下面介绍两种 Web 应用功能测试的自动化技术，一个是 Web 应用链接质量保证技术，另一个是 Web 应用功能测试技术，下面分别论述。

1. Web 应用链接质量保证技术

链接是使用户从一个页面浏览到另一个页面的重要手段，链接的质量决定着功能是否能够成功实现。

要保证每个链接的质量，需要做好三件事情：

① 该链接将用户带到它所说明的地方；

② 被链接页面是存在的；

③ 保证 Web 应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面。

链接测试非常复杂。比如当网页的结构非常复杂且数量巨大时，链接检查的速度就迫切需要提高。当网络连接总是不稳定的时候，误判的频率增大导致工作量加大，就需要保证工作进度。还有，测试的结果能否清晰地报告出来等，这些需求都提高了测试的复杂度。

要测试 Web 应用的链接，可以借助于自动化的 Web 应用链接测试工具，例如 WebCheck、Linkbot、TestPartner 等。这些测试工具在测试过程中自动扫描 Web 应用的所有链接，定位及报告问题。针对应用中存在的各种各样的链接，比如图片、框架（Frame）、插件（Plugin）、背景、样式表（Style Sheet）、脚本、Java Applet 等以及支持的连接种类，比如 HTTP、FTP、GOPHER、HTTPS 等工具都能够支持。另外，对本地

的链接和重定向的链接也能很好地支持。例如 WebCheck 能够定位约 50 个的问题类型，并且提供 19 个 HTML 格式的报告。

利用自动化测试工具测试 Web 应用的链接，主要优势体现在以下几个方面。

- 简单易用；
- 在实现上采用多线程技术，因此检查速度特别快；
- 对断开的连接可以再次检查，避免误判；
- 没有检查连接的数量限制，只受系统资源的约束；
- 可以分析 Web 应用的结构；
- 检查结果可以分类查看，自动生成 HTML 格式的报告。

Web 应用链接主要测试点如下。

- 测试内部和外部链接中成功和失败的链接点，以及应用中不被其他链接调用的页面；
- 测试链接中新网页、老网页、慢网页以及丢失的图象标题标签和属性标签等；
- 分析 Web 应用的结构是否合理，包括显示和某个 URL 相关的链接及按照标题、描述、作者、大小、最后修改时间、类型为 URL 链接分类等。

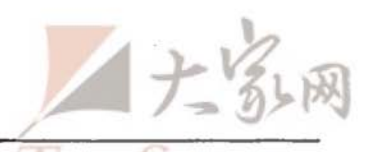
2．Web 应用功能测试技术

如果开发人员刚创立一个新的 Web 应用系统。在发布应用系统之前，它必须经过测试以确保一切设定功能都能正常运行，这样的测试任务中，针对同一模块或者同一功能点的测试可能需要重复多次。另外，在一个公司中不同项目的测试可能并行展开，例如人事部门的 HR 系统、客服部门的 CRM 系统、物流部门的 ERP 系统等。这样的现状就会使测试人员面临这样一个问题，即“如何有效地测试不断修改着的一个或多个应用程序”。如果资源有限的话，这个问题就更加棘手。人工测试的工作量太大，况且很多公司负担不起额外的时间来培训新的测试人员。为了解决这个问题，就需要一个能简单操作的测试工具来自动完成功能性测试。

Mercury Interactive 的 WinRunner 就是一个功能性测试工具。它通过捕获和重放用户对 Web 应用程序的操作，WinRunner 可自动执行功能性测试。下面我们来看一个标准的测试过程，主要步骤包括：创建测试脚本、插入检查点、运行测试以及分析结果。

- 创建测试脚本。

创建测试脚本只需记录下一个标准的业务流程，如下一张订单或建立一个新的商家账户。测试人员在 GUI 上点击鼠标，测试工具记录流程就可建立测试脚本，即使技术知识有限的用户也能生成完整的测试。脚本可以直接编辑来满足各种复杂测试的需求。例如，WinRunner 可以将两种测试脚本创建方式结合在一个环境下，来适应测试需求。这两种测试脚本创建方式分别是模拟控件操作和模拟鼠标操作。



- 插入检查点。

在记录一个测试的过程的脚本中，测试工程师可插入检查点，测试工具会收集检查点的性能指标。脚本运行时，测试工具在查寻潜在错误的同时，会比较检查点所设定的结果和实际测试结果，对其一一验证。例如，WinRunner 允许您使用几种不同类型的检查点，包括文本、GUI、位图和数据库。用一个位图检查点，可以确认一个位图图像，如公司的图标是否出现在指定位置。

- 运行测试。

建立起测试脚本，并插入检查点和做一些必要修改后，就可以开始运行测试。当测试工具执行测试时，它会自动操作应用程序，正如一个真实用户根据记录流程执行着每一步的操作。

- 分析结果。

一旦测试结束后，就需要分析测试结果。测试工具一般会提供详尽的、易读的报告，这些报告对在测试运行中发生的重要事件进行描述，如出错内容和检查点等。

一次测试结束后，随着时间推移，开发人员会对应用程序做进一步的修改，并需要另加额外的测试。有了前面利用自动化测试工具进行测试的基础，不必改动测试脚本，就可以重新建一个新的测试，这样大大地节省了时间和资源，充分利用了测试投资。

功能自动化测试工具还能验证数据库的数值，从而确保交易的准确性。例如，在创建测试脚本时，可以设定哪些数据库表格和记录资料需要检测。在重放时，测试程序就会核对数据库内的实际数值与脚本中设定的数值，在有更新/修改，删除或插入的记录上会使用突出标识以引起注意。

有时为了彻底全面地测试一个应用程序，需要了解对于不同类型的数据，它是如何运行的。测试工具可以将一个记录下的业务流程转化为一个数据驱动的测试，来反映多个用户各自独特且真实的操作行为。以一个订单输入的流程为例，测试人员或许希望将一些锁定的项目栏，如定单号或客户名转化为可变栏，这样就可以用多套数值来检测应用程序了。数据来源可以采用自动生成表格，也可直接从其他的表格或数据库中导入。数据驱动性测试不仅节省了时间和资源，而且提高了应用程序的测试覆盖率。

利用自动化测试工具在对脚本进行编辑的时候，可以从列表里选择一个功能函数加到脚本中，以提高测试能力。例如，点击“calendar”，然后从年历功能中的下属目录中选择，如“calendar_select_date()”，工具会提供函数的解释。选定了这个函数后，可以输入参数，再将这个函数插入到测试脚本中。

9.4.3 易用性测试

使用 Web 浏览器作为客户端的一个原因就是它易于使用。用户知道如何浏览一个构

建良好的网站。如果你注重这方面的测试，那么验证应用程序是否易于使用就非常重要了。

要评估一个 Web 系统是否易用，首先要分析最终用户的情况。用户的情况决定了 Web 系统在易用性方面需要花费多少时间，以及易用设计的方向。从用户角度讲，我们主要考虑以下几个方面的情况。

- 用户的计算机使用经验；
- 用户对浏览器以及 Web 的使用经验；
- 用户的业务专业知识。

确定了最终用户使用的基础情况，我们就可以有针对性地测试一个 Web 系统的易用性了。我们把 Web 系统的易用性测试分为三个方面进行测试：

- 界面测试；
- 辅助功能测试；
- 图形测试。

下面我们将对此进行详细的叙述。

1．界面测试

有人说，除了一定的技术外，Web 系统的好与坏取决于其页面的设计艺术水平。这个话虽有失偏颇，但也从另外一个角度说明了 Web 系统的界面的重要性。由于 Web 系统的客户端均采用浏览器，不同用户可能会在浏览器里设置不同的显示方式，因此，我们在作界面测试的时候尽量使用默认的设置。

在开始进行界面测试以前，我们需要重点调研两个问题：

- Web 应用系统的最终用户群是谁；
- Web 应用界面（大多数等同于网页）的设计策略是什么。

这两个问题决定了我们评价一个 Web 应用系统界面采用什么样的标准。尽管不同的 Web 应用系统的界面千变万化，但其测试方法和评价准则仍有一些共同的内容。以下是界面测试中需要重点关注的。

- 页面中各元素布局的协调性。

一个复杂的页面会包含多种元素，如文字、表单、图片、动画、表格、视频等，从美学的角度来看，如果只是把所有的 Web 应用功能简单地堆砌到页面上，其易用性是很差的。需要评估的协调性包括以下几个方面：

① 各元素位置的协调性；

② 各元素颜色的协调性；

③ 各元素大小比例的协调性。

- 不同页面风格的统一性。



一个统一的Web系统的不同页面应该从颜色、框架、操作方式等多个方面统一起来。由于Web日益流行，很多人把它看作图形设计作品。人们为了体现出网页设计的丰富多彩，在不同页面使用很多不同风格的图片、特效，初看觉得网页非常艳丽多彩，但忽略了不同网页的协调统一。

- 用户在界面中操作的便利性。

网页设计中用得最多的是表格，而用户操作最容易导致界面缺陷的也就是表格，因此要对表格进行操作验证。例如，需要验证表格是否设置正确、用户是否需要向右滚动页面才能看见表格中的内容；每一栏的宽度是否足够宽，表格里的文字是否都有折行；是否有因为某一格的内容太多，而将整行的内容拉长等。

- 界面动态操作测试。

主要测试以下几个方面：

① 屏幕分辨率设置的影响；

② 浏览窗口最大化/最小化的影响；

③ 选定目标元素的置中与缩放。

2．辅助功能测试

辅助功能指为了方便用户更快更容易地使用网站而采用的一些设置，主要包括使用说明、导航、站点地图、帮助等。

① 使用说明。应该确认站点有使用说明。即使网站很简单，也可能有人在某些方面需要证实一下。测试人员需要测试说明文档，验证说明是正确的。还可以根据说明进行操作，确认出现预期的结果。

② 导航。导航描述了用户在一个页面内操作的方式，在不同的用户接口控制之间，例如按钮、对话框、列表和窗口等；或在不同的连接页面之间。通过考虑下列问题，可以决定一个Web应用系统是否易于导航：导航是否直观，Web系统的主要部分是否可通过主页存取。

在一个页面上放太多的信息往往起到与预期相反的效果。Web应用系统的用户趋向于目的驱动，很快地扫描一个Web应用系统，看是否有满足自己需要的信息，如果没有，就会很快地离开。很少有用户愿意花时间去熟悉Web应用系统的结构，因此，Web应用系统导航帮助要尽可能地准确。

导航的另一个重要方面是Web应用系统的页面结构、导航、菜单、连接的风格是否一致。确保用户凭直觉就知道Web应用系统里面是否还有内容，内容在什么地方。

③ 站点地图。确认你测试的站点是否有地图。有些网络高手可以直接去自己要去的地方，而不必点击一大堆页面。另外，新用户在网站中可能会迷失方向。站点地图可以引导用户进行浏览。需要验证站点地图是否正确，确认地图上的链接是否确实存，地

图有没有包括站点上的所有链接。

④ 帮助。帮助在软件的易用性中占很重要的位置，在 Web 系统中也不例外。Web 系统中的帮助可以分为联机帮助、参考式帮助、教程式帮助等，可分别针对其主题进行测试。

3．图形测试

我们这里谈到的图形，泛指页面中所有的图片以及彩色元素。在 Web 应用系统中，图形占有相当重要的位置，因为 Web 系统应用范围的特殊性，图形直接影响了用户浏览和使用 Web 系统的操作。

适当的图片和动画既能起到广告宣传的作用，又能起到美化页面的功能。和谐的颜色搭配既能体现出 Web 应用系统的主题，也能反映出业主企业的形象。一个 Web 应用系统的图形可以包括图片、动画、边框、颜色、字体、背景、按钮，我们在进行图形测试时重点进行以下测试。

- 验证所有的图形是否有明确的用途。特别是图片或动画，有无堆砌的现象，因为过于集中的图片，既浪费传输时间，又影响视觉效果。
- 验证所有页面字体的颜色、风格是否一致。字体的颜色应与页面的主色调是否协调。
- 背景颜色与字体颜色和前景颜色是否相搭配。
- 确认图片的大小和质量。这也是一个很重要的因素，图片是否采用 JPG 或 GIF 压缩，图片尺寸是否合适，分辨率是否满足需求等都需要进行评价。

9.4.4 负载压力测试

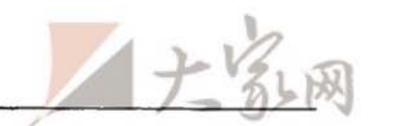
性能是用户经常会遇到的一个棘手的问题，也可能是 Web 系统在投入实际使用以前最为关心的问题。

Web 系统的性能包含哪些方面呢？

- 客户端向服务器发出一个请求。
- 服务器分配请求并进行处理。
- 服务器把处理的结果反馈给客户端。
- 客户端对结果进行分析，显示出来或进一步执行。

从这个过程可以看出，由于客户端是一个单独的个体，几乎不会出现性能问题，而服务器为了响应多个客户端的请求，有可能出现响应错误、响应缓慢、数据丢失等错误。

用户连接到 Web 应用系统的速度根据上网方式的变化而变化，他们或是电话拨号，或是宽带上网。当下载一个程序时，用户可以等较长的时间，但如果仅仅访问一个页面就不会这样。如果 Web 系统响应时间太长，用户就会因没有耐心等待而离开。另外，有



些页面有超时的限制，如果响应速度太慢，用户可能还没来得及浏览内容，就需要重新登录了。而且，连接速度太慢，还可能引起数据丢失，使用户得不到真实的页面。

负载测试是为了测量 Web 系统在某一负载级别上的性能，以保证 Web 系统在需求范围内能正常工作。负载级别可以是某个时刻同时访问 Web 系统的用户数量，也可以是在线处理的数据量。例如，Web 应用系统能允许多少个用户同时在线；如果超过了这个数量，会出现什么现象；Web 应用系统能否处理大量用户对同一个页面的请求。负载测试应该安排在 Web 系统发布以后，在实际的网络环境中进行测试。

进行压力测试是指实际破坏一个 Web 应用系统，测试系统的反映。压力测试是测试系统的限制和故障恢复能力，也就是测试 Web 应用系统会不会崩溃，在什么情况下会崩溃。

一般压力测试包含如下步骤：

- 确定交易执行响应时间。
- 估计 Web 系统能够承受的最大并发用户数量。
- 模拟用户请求，以一个比较小的负载开始，逐渐增加模拟用户的数量，直到系统不能承受负载为止。
- 如果负载没有达到需求，那么应该优化这个 Web 程序。

9.4.5 客户端配置与兼容性测试

1．配置测试与兼容性测试概述

要实现完全的配置测试和兼容性测试是不可能的，而且从测试成本的角度来看也是不可取的。本项测试目的，是发现 Web 系统在可能的用户环境下运行程序时出现的错误，对什么样的用户环境进行配置测试与兼容性测试，是需要我们进行分析与调查的。由于 Web 系统采用浏览器/服务器的模式，我们把配置测试与兼容性测试的着重点放在客户端。而客户端最重要的两个因素就是浏览器与操作系统，所以面向用户的配置测试与兼容性测试可分为以下三个方面：

① 浏览器的配置测试；

② 平台兼容性测试；

③ 浏览器兼容性测试。

2．浏览器的配置测试

前面提到，浏览器中有许多会影响 Web 功能的设置，例如缓存设置、cookies 设置、显示设置、安全设置等，在完成了功能测试后，我要需要对选用的浏览器进行配置测试，也就是测试不同配置对 Web 功能的影响程度，再核查有影响的配置在功能说明书中是否有明确提示。

3. 平台兼容性测试

市场上有很多不同的操作系统类型，最常见的有 Windows、UNIX、Macintosh、Linux 等。Web 应用系统的最终用户究竟使用哪一种操作系统，取决于用户系统的配置。这样，就可能会发生兼容性问题，同一个应用可能在某些操作系统下能够正常运行，但在另外的操作系统下可能会运行失败。

因此，在 Web 系统发布之前，需要在用户可能用到的操作系统下，对 Web 系统进行兼容性测试。

4. 浏览器兼容性测试

浏览器是 Web 客户端最核心的构件，来自不同厂商的浏览器对 Java、Javascript、ActiveX、Plug-ins 或不同的 HTML 规格有不同的支持。例如，ActiveX 是 Microsoft 的产品，是为 Internet Explorer 而设计的，Javascript 是 Netscape 的产品，Java 是 Sun 的产品等。另外，框架和层次结构风格在不同的浏览器中也有不同的显示，甚至根本不显示。不同的浏览器对安全性和 Java 的设置也不一样。

测试浏览器的兼容性可以与操作系统的兼容性结合起来，最有效的方法是创建一个兼容性矩阵，如表 9-2 所示。在这个矩阵中，测试不同版本操作系统上的不同厂商、不同版本的浏览器对某些构件和设置的适应性。

表 9-2　兼容性测试矩阵

浏览器 操作系统	IE5	IE5.5	IE6.0	…
Windows 98				
Windows 2000 Pro				
Windows XP Pro				
Windows 2000 Ser				
…				

9.4.6 安全性测试

测试应用程序的体系结构和设计可以消除很多与设计有关的漏洞，从而提高应用程序的整体安全性。设计时修复漏洞要比在开发后期解决问题更为简单，也更经济，因为开发后期可能要进行大量的重新工程处理。开发时如果考虑一些与目标部署环境相关的设计以及该环境定义的安全策略，可确保应用程序的部署更加平稳和安全。如果应用程序已创建完毕，安全测试可修复漏洞并完善未来的设计。

一个完整的 Web 安全体系测试可以从部署与基础结构、输入验证、身份验证、授权、配置管理、敏感数据、会话管理、加密、参数操作、异常管理、审核和日志记录等几个

方面入手。

1．安全体系测试

（1）部署与基础结构

检验底层网络和主机基础结构提供给应用程序的安全设置，然后检验运行环境要求的所有限制。此外，考虑部署的拓扑结构以及中间层应用程序服务器、外围区域以及内部防火墙对设计的影响。检验下列问题，确定可能存在的部署和基础结构问题。

- 网络是否提供了安全的通信。

数据在客户端与服务器（或服务器与服务器）之间传输时最易受到攻击。网络负责数据传输的完整性和私密性。如果必须保证数据安全，可使用适当的加密算法。此外，还必须确保网络设备安全。因为这是维护网络完整性所必需的。

- 部署拓扑结构是否包括内部防火墙。

如果内部防火墙将 Web 服务器与应用程序服务器（或数据库服务器）分隔开来，则需要考虑下列问题，确保设计能适应这种配置。

① 下游服务器如何验证 Web 服务器的身份；

② 如果使用域账户和 Windows 身份验证，防火墙是否打开了必要的端口；

③ 是否使用分布式事务；

④ 如果 Web 服务器使用 DTC（Microsoft Distributed Transaction Coordinator）的服务来启动分布式事务，内部防火墙是否为 DTC 通信打开了必要的端口。

- 部署拓扑结构中是否包括远程应用程序服务器。

如果部署拓扑结构包括了一个物理远程中间层，则需要考虑下列问题。

① 是否使用企业服务。如果是，是否已限制了 DCOM 端口范围，内部防火墙是否打开了这些端口。

② 是否使用.NET 远程处理。

③ 远程处理用在受信服务器方案中，网络是否支持 IPSec 策略。

④ ASP.NET 承载远程组件是否支持身份验证和授权。

⑤ 是否使用 Web 服务，如果是，中间层 Web 服务如何验证 Web 应用程序的身份。Web 应用程序是否通过在 Web 服务代理中配置凭据来使 Web 服务验证 Web 服务器的身份，如果否，Web 服务如何明确调用者。

- 基础结构安全性要求的限制是什么。

设计是否假定主机基础结构安全限制要失效。例如，安全限制可能要求根据所需的服务、协议或账户特权来对设计进行权衡。需要考虑下列问题。

① 是否依赖可能不可用的服务或协议。开发和测试环境中可用的服务和协议可能在生产环境中不可用。

② 是否依赖敏感的账户特权。设计应尽量使用特权最少的进程、服务和用户账户。

③ 要执行的操作是否要求可能不被许可的敏感特权。例如，应用程序是否要创建线程级模拟令牌来创建资源访问的服务身份。这项操作要求“作为操作系统的一部分”特权，而该特权不应授予 Web 服务器进程（因为可能增加进程被利用的风险）。如果需要此功能，设计应对更高级别的特权进行划分，例如，在进程外的企业服务应用程序中。

- 目标环境支持怎样的信任级别。

运行环境的代码访问安全信任级别决定了代码可访问的资源，以及它能执行的特权操作。请检查运行环境支持的信任级别。如果允许 Web 应用程序以完全信任级别运行，代码将能够访问操作系统安全性许可的任何资源。

如果 Web 应用程序必须以受限信任级别运行，则代码能访问的资源类型以及能执行的特权操作都将受到一定的限制。在部分信任案例中，设计应对特权代码进行沙盒（sandboxing）处理。此外，还应使用不同的程序集来分隔特权代码。这样，可以对特权代码和应用程序的其余部分单独配置特权代码，然后授予必要的附加代码访问权限。

注意：如果应用程序部署在共享服务器（或应用程序将由宿主公司运行），信任级别通常是个问题。此时，需要检查安全策略，然后确定 Web 应用程序的信任级别。

（2）输入验证

需要对应用程序验证输入内容的方式进行检验，因为很多 Web 应用程序攻击都故意使用格式错误的输入。SQL 注入、跨站点脚本（XSS）、缓冲区溢出、代码注入以及无数其他拒绝服务和特权提升攻击都可利用输入验证中的漏洞。表 9-3 中重点列出了常见的输入验证漏洞。

表 9-3　常见的输入验证漏洞

漏　　洞	影　　响
超文本标记语言（HTML）输出流中有未经验证的输入	应用程序易受 XSS 攻击
用于生成 SQL 查询的输入未经验证	应用程序易受 SQL 注入攻击
依赖客户端的验证	客户端验证很容易被忽略
使用输入文件名、URL 或用户名制定安全决策	应用程序易出现规范化错误并导致安全漏洞
对恶意输入仅采用应用程序筛选器	这基本上不可能达到目的，原因是可能的恶意攻击范围太大。应用程序应对输入进行约束、拒绝和净化

测试时应考虑下列问题，以帮助发现潜在的输入验证安全问题。

- 如何验证输入。

设计指定的输入验证方法是什么？首先，设计必须展示策略。应用程序应对收到的



所有输入进行约束、拒绝和净化。约束输入是最佳的方法，因为针对已知有效类型、模式和范围对数据进行验证，要比通过查找已知坏字符来验证数据简单得多。

测试时应考虑下列问题，帮助识别潜在的漏洞。

① 是否清楚入口点。

确保设计标出了应用程序的入口点，以便跟踪各个输入字段的操作。可考虑 Web 页输入、输入到组件和 Web 服务，以及从数据库输入。

② 是否清楚信任边界。

如果输入是从信任边界内受信源传递的，并非总要验证输入；但如果输入是从不受信任的源传递的，必须验证输入。

③ 是否验证 Web 页输入。

不要将最终用户看作受信任的数据源。确保对正常和隐藏的表单字段、查询字符串和 cookie 都进行验证。

④ 是否对传递到组件或 Web 服务的参数进行验证。

如果不进行验证，惟一的安全条件就是数据接收自当前信任边界之内。但是，如果使用深层防御策略，则需要使用多层验证。

⑤ 是否验证从数据库中检索的数据。

这种形式的输入也应验证，特别是当其他应用程序也写入该数据库时。不要对其他应用程序的数据验证程度进行假设。

⑥ 是否将方法集中起来。

对于相同类型的输入字段类型，检验使用的是否是相同的验证和筛选库，确保一致地执行验证规则。

⑦ 是否依赖客户端的验证。

客户端验证可用于降低到服务器的回程数量，但不能依靠它来维护安全性，因为它很容易被忽略。需要在服务器验证所有的输入。

- 如何处理输入。

检查应用程序处理输入的方式，不同类型的处理可能导致不同类型的漏洞。例如，如果在 SQL 查询中使用输入，应用程序可能易受 SQL 注入的攻击。

测试中考虑下列问题，帮助发现潜在的漏洞。

① 应用程序是否易受规范化问题的影响。

检查应用程序是否使用基于输入的名称来制定安全决策。例如，应用程序是否接受用户名、文件名或 URL。由于名称的表示方法多种多样，以上各项都极易造成规范化错误问题。如果应用程序接受输入作为名称，则应确保对它们进行验证并在处理之前将它们转换为规范的表示法。

② 应用程序是否易受 SQL 注入攻击。

密切注意形成 SQL 数据库查询的所有输入字段。确保对这些字段的类型、格式、长度和范围进行正确的验证。此外，检查查询的生成方式。如果使用参数化的存储过程，输入参数将被当作文本，而不会当作可执行代码。这是降低风险的一种有效措施。

③ 应用程序是否易受 XSS 攻击。

如果在 HTML 输出流中包括输入字段，可能受到 XSS 攻击。确保对输入进行验证，并对输出进行编码。密切注意系统对接受一定范围 HTML 字符的输入字段的处理方法。

（3）身份验证

检查应用程序验证调用者身份的方法，在何处使用身份验证，如何确保凭据在存储中或通过网络传递的安全。身份验证中的漏洞可能导致应用程序易受哄骗攻击、词典攻击、会话劫持等。表 9-4 重点列出了常见的身份验证漏洞。

表 9-4　常见的身份验证漏洞

漏　洞	影　响
弱密码	增加了破解密码和词典攻击的风险
配置文件中使用明文凭据	可访问服务器的内部人员（或利用主机漏洞下载配置文件的攻击者）都能直接访问该凭据
通过网络传递明文凭据	攻击者可通过监控网络来盗取身份验证凭据并窃取身份
账户的特权过强	与进程和账户泄露相关的风险增加
长会话	与会话劫持相关的风险增加
混用个性化数据的身份验证数据	个性化数据适于永久的 cookie，而身份验证 cookie 不应是永久的

测试中需要考虑下列问题，确定在应用程序进行身份验证的方法中的潜在漏洞。

- 是否区分公共访问和受限访问。

如果应用程序既有不要求身份验证的公共区域，也有要求身份验证的受限区域，检查站点设计区分二者的方法。必须为受限的页和资源使用单独的子文件夹，然后在 IIS 中将它们配置为要求 SSL 来确保安全。这种方法允许只在需要的地方使用 SSL 来确保敏感数据和身法验证 cookie 的安全性，从而避免了因在整个站点中使用 SSL 而造成的附加性能负担。

- 是否明确服务账户要求。

设计应明确连接不同资源（包括数据库、目录服务和其他类型的网络资源）的服务账户范围。设计中不能使用单个的、有高度特权的账户（有足够的权限连接多种不同类型的资源）。

① 设计是否要求特权最少的账户。



检查设计并准确标识各账户执行特定功能所需的特权，然后在任何情况下都使用特权最少的账户。

② 应用程序是否要维护服务账户凭据。

如果是，确保加密这些凭据，然后保存在受限的位置中。例如，保存在有受限访问控制列表（ACL）的注册表项。

- 如何验证调用者身份。

测试时考虑与调用者身份验证相关的下列事项。具体事项由设计中使用的身份验证类型决定。

① 是否在网络中传递明文凭据。如果使用表单或基本身份验证（或使用 Web 服务并在 SOAP 头中传递凭据），确保使用 SSL 来保护传输中的凭据。

② 是否实现自己的用户存储。如果是，检查用户凭据的存储位置和存储方式。一种常见错误是将明文或加密密码保存在用户存储中。实际上，必须保存密码的哈希值来进行身份验证。

如果根据 SQL Server 用户存储验证凭据，密切注意用户名和密码的输入。检查是否存在恶意注入的 SQL 字符。

③ 是否使用表单身份验证。如果是，除使用 SSL 保护凭据外，还应使用 SSL 来保护身份验证 cookie。此外，还要检查设计是否使用受限的会话生存期来抵御 cookie 重播攻击，并确保加密 cookie。

- 如何验证数据库的身份。

如果应用程序要连接数据库，检查使用的身份验证机制、打算使用的账户（一个或多个），以及如何在数据库中授权应用程序。

明确下列问题有助于对数据库身份验证进行评价。

① 是否使用 SQL 身份验证。

在理想情况下，设计使用 Windows 身份验证来连接 SQL Server，因为这种方法本身更加安全。如果使用 SQL 身份验证，检查在网络中和数据库连接字符串中确保凭据安全的方法。

如果网络基础结构不提供 IPSec 加密通道，确保在数据库中安装服务器证书来提供自动 SQL 凭据加密。此外，还要检验确保数据库连接字符串安全的方法，因为这些字符串中包含 SQL 账户的用户名和密码。

② 是否使用进程账户。

如果使用应用程序的进程账户并使用 Windows 身份验证连接 SQL 服务器，应在设计中使用特权最少的账户。本地 ASP.NET 账户便是为此提供的，尽管对于本地账户来说，用户需要在数据库服务器上创建一个相同的账户。

如果打算使用域账户，首先确保它是特权最少的账户，然后打开相关的端口来确保所有相关防火墙都支持 Windows 身份验证。

③ 是否使用服务账户。

如果设计要求使用多个身份来支持数据库中的高粒度授权，则需要检查保存账户凭据（在理想情况下，这些凭据使用数据保护 API（DPAPI）加密并保存在安全注册表项中）的方法，以及使用服务身份的方法。

此外，还要检查使用哪些进程通过该服务账户创建模拟的安全上下文。该操作不应由 Microsoft Windows 2000 中的 ASP.NET 应用程序进程来完成，因为它将强制提升进程账户的特权，并授予“作为操作系统的一部分”特权。这种情况必须尽量避免，它将大大增加风险。

④ 是否考虑使用匿名 Internet 用户身份。

对于使用表单或 Passport 身份验证的应用程序而言，可为各个程序配置单独的匿名用户账户。然后，启用模拟并使用匿名身份来访问数据库。该方法适于对同一服务器的不同应用程序进行单独的授权和身份跟踪。

⑤ 是否使用原始用户身份。

如果设计要求模拟原始调用者，必须考虑该方法是否能提供足够的伸缩性，因为连接池是无效的。另一种备选方法是，通过受信的查询参数在应用程序级流动原始调用者身份。

⑥ 如何保存数据库连接字符串。

如果数据库连接字符串硬编码，或以明文形式保存在配置文件或 COM+ 目录中，则很容易受到攻击。实际上，应加密它们，然后限制对加密数据的访问。

- 是否强制使用强账户管理措施。

如果应用程序使用 Windows 身份验证，Windows 安全策略将强制使用强密码、受限登录和其他最佳账户管理策略。其他情况，则由应用程序层负责这些措施。测试要考虑与应用程序账户管理相关的下列问题。

① 应用程序是否强制使用强密码。

例如，ASP.NET Web 页是否使用正则表达式来验证密码复杂性规则。

② 是否限制失败登录的次数。

这样做有助于对抗词典攻击。

③ 是否在故障发生后公开过多的信息。

确保不显示类似“不正确的密码”这样的消息，因为它将告诉恶意用户：用户名是正确的。结果，恶意用户便可集中精力破解密码。

④ 是否强制定期更改密码。



如果不强制定期更改密码，用户极有可能不更改自己的密码，结果风险更高。

⑤ 是否能在泄露发生时迅速禁用账户。

如果账户泄露，是否能方便地禁用账户来防止攻击者继续使用账户。

⑥ 应用程序是否记录登录企图。

记录失败的登录企图是检测攻击者试图侵入的有效方法。

（4）授权

检查应用程序是如何向用户授权的。还要检验应用程序在数据库中是如何被授权的，以及如何控制系统级资源的访问。授权中的漏洞可能导致信息泄漏、数据篡改及特权提升。使用深层防御策略是一种重要的方法，它可应用于应用程序的授权策略中。表 9-5 重点列出了常见的授权漏洞。

表 9-5 常见的授权漏洞

漏 洞	影 响
依赖单个网关守卫	如果网关守卫被忽略或配置不正确，用户能不经授权进行访问
不能根据应用程序身份锁定系统资源	攻击者可强制应用程序访问受限的系统资源
不能将数据库访问限定于特定存储过程	攻击者可利用 SQL 注入攻击来检索、操作或毁坏数据
特权隔离不充分	没有责任或能力对每个用户进行授权

测试中需要考虑下列问题，帮助验证应用程序设计的授权策略。

- 如何向最终用户授权。

应在设计时从两种角度考虑授权。首先，考虑最终用户授权。哪些用户可访问哪些资源，并执行哪些操作。其次，如何防止恶意用户使用应用程序访问系统级资源。考虑下列问题，验证应用程序的授权策略。

① 是否使用深层防御策略。

确保设计不依赖于单个网关守卫来加强访问控制。考虑该网关守卫失败（或攻击者设法忽略它）时发生的情况。

② 使用了哪些网关守卫。

可能的选择有 IISWeb 权限、NTFS 权限、ASP.NET 文件授权（仅适用于 Windows 身份验证）、URL 授权和用户权限请求。如果不使用某个特定类型，需要明确不使用的理由。

③ 是否使用基于角色的方法。

如果是，如何维护角色列表，维护角色列表所需的管理界面安全性如何。

④ 角色是否提供足够的特权隔离。

设计是否提供了适当的粒度，使不同用户角色的关联特权得到充分的隔离。避免出现仅为满足特定用户需要而授予角色较高特权的情况。

- 如何在数据库中授权应用程序。

在应用程序中连接数据库的账户必须有受限的能力，只需满足应用程序的要求即可，不要再高了。

应用程序是否使用存储过程来访问数据库呢？建议应用程序使用存储过程来访问数据库，因为一般用户只能授予应用程序登录访问特定存储过程的权限。可以限制登录不在数据库中直接执行创建/读取/更新/删除（CRUD）操作。

- 如何将访问限定于系统级资源。

设计应用程序时，应考虑应用程序在可访问系统级资源方面的限制。只能授予应用程序访问最低限度的资源。这是缓解风险的一种策略，可在应用程序遭受攻击时限制受损程度。考虑下列问题。

① 设计是否使用代码访问安全性。

代码访问安全性提供了一种资源约束模型，该模型可防止代码（和 Web 应用程序）访问特定类型的系统级资源。如果使用代码访问安全性，设计必将受到影响。明确是否在设计规划中包括代码访问安全性，然后通过对特权代码进行隔离和沙盒处理（sandboxing），并将资源访问代码置于自己独立的程序集中，从而进行相应的设计。

② 应用程序使用哪些身份。

设计必须明确应用程序使用的所有身份，包括进程身份和所有模拟身份（包括匿名 Internet 用户账户和服务身份）。此外，设计还要指出这些身份要访问的资源。

在部署时，可对系统级资源配置正确的 ACL，确保应用程序的身份只能访问所需的资源。

（5）配置管理

如果应用程序提供了可配置的管理界面，要检查确保管理界面安全的方法。此外，还要检查如何确保敏感配置数据的安全。表 9-6 显示了常见的配置管理漏洞。

表 9-6　常见的配置管理漏洞

漏　　洞	影　　响
不安全的管理界面	未经授权的用户可重新配置应用程序，并访问敏感数据
不安全的配置存储	未经授权的用户可访问配置存储并获取机密信息（如账户名、密码和数据库连接详细信息）
明文配置数据	可登录服务器的所有用户都能查看敏感的配置数据
管理员太多	这使管理员的审核和评价工作变得很困难
进程账户和服务账户的特权过高	这可导致特权提升攻击



测试时考虑下列问题，帮助验证应用程序设计在配置管理方面的方法。

- 是否支持远程管理。

如果设计指定了远程管理，必须确保管理界面和配置存储的安全，因为这些操作本身非常敏感，而且通过管理界面访问的数据也很敏感。考虑与远程管理设计相关的下列问题。

① 是否使用强身份验证。

必须要求对所有管理界面用户进行身份验证。使用强身份验证，如 Windows 或客户端证书身份验证。

② 是否加密网络通信数据。

使用经过加密的信道，如 IPSec 或虚拟专用网络（VPN）连接提供的通道。不支持不安全通道中的远程管理。IPSec 允许对可用来管理服务器的客户计算机的身份和数量进行限制。

- 是否保证配置存储的安全。

明确应用程序的配置存储，然后检查限制访问这些存储的方法，以及确保存储中数据安全的方法。

① 配置存储是否在 Web 空间中。

对于保存在 Web 空间文件中的配置数据，其安全性要低于保存在 Web 空间之外的数据。主机配置错误或未发现的 Bug 都可能导致攻击者通过 HTTP 检索，并下载配置文件。

② 配置存储中的数据是否安全。

确保在存储中加密关键的配置数据项（如数据库连接字符串、加密密钥和服务账户凭据）。

③ 如何限制对配置存储的访问。

确保管理界面提供必要的授权，只有经过验证的管理员才可访问并操作这些数据。

- 是否隔离管理员特权。

如果管理界面支持不同的功能（如站点内容更新，服务账户重新配置和数据库连接详细信息），要确认管理界面支持基于角色的授权，从而区分内容开发人员和操作员或系统管理员。例如，不必许可更新静态 Web 站点的人改变客户的信用额度或重新配置数据库连接字符串。

（6）敏感数据

检查应用程序对存储中、应用程序内存中以及网络中的敏感数据的处理方法。表 9-7 显示了与处理敏感数据相关的常见漏洞。

表 9-7　敏感数据处理中的常见漏洞

漏　洞	影　响
在不必要的情况下存储机密信息	与不存储机密信息相比，这首先大大增加了安全风险
在代码中存储机密数据	如果代码位于服务器，攻击者可能下载它。在二进制程序集中，机密信息是可见的
以明文形式存储机密信息	任何能登录服务器的人都可看到机密数据
在网络中以明文传递敏感数据	窃听者可通过监控网络来获取并篡改这些数据

测试中要考虑下列问题，帮助验证应用程序处理敏感数据的方法。

- 是否存储机密信息。

机密信息包括了应用程序的配置数据，如账户密码和加密密钥。如果可能，明确其他避免保存机密信息的设计方法。如果要处理机密信息，由系统平台处理它们，尽可能不在应用程序中承担这一任务。如果确实要保存机密信息，则要考虑下列问题。

① 是否能避免存储机密信息。

如果使用其他的实施技术，是否能避免存储机密信息。例如，如果只需了解用户是否知道密码，则无需存储密码。或者，仅保存单向的密码哈希值。

此外，如果使用 Windows 身份验证，可通过嵌套凭据来避免存储连接字符串。

② 如何存储机密信息。

如果使用加密，如何确保加密密钥的安全。考虑使用系统平台提供的 DPAPI 加密，这种加密能替用户完成密钥管理。

③ 在何处保存机密信息。

检查应用程序保存加密数据的方法。要获得尽可能高的安全性，应使用 Windows ACL 限制对加密数据的访问。确认应用程序不以明文或源代码形式存储机密信息。

如果使用本地安全机构（LSA），检索机密信息的代码必须使用管理员特权才可以运行，这将增加风险。另一种不要求扩展特权的方法是使用 DPAPI。

④ 如何处理机密信息。

检验应用程序访问机密信息的方法，以及它们以明文形式存留在内存中的时间。机密信息通常应根据需要检索，并尽快使用，然后丢弃。

⑤ 是否在 cookie 中存储机密信息。

如果是，应确保 cookie 是加密的，且不会永久保存在客户计算机中。

- 如何存储敏感数据。

如果存储了敏感的应用程序数据（如客户信用卡详细信息），检查数据保护方法。

① 使用怎样的加密算法。

应使用强加密算法来加密。例如，使用较长的密钥（如 Triple DES）。



② 如何确保加密密钥的安全性。

数据的安全性与加密密钥安全性同等重要。因此，检查确保密钥安全的方法。在理想状况下，使用 DPAPI 加密密钥并保存在受限位置（如注册表项中）来确保安全。

- 是否在网络中传递敏感数据。

如果通过网络传递敏感数据，应确保通过应用程序加密这些数据，或通过加密的通信链接来传递它们。

- 是否记录敏感数据。

检查应用程序（或主机）是否在明文日志文件中记录用户账户密码这样的敏感数据。通常，必须避免这样做。确保应用程序不在查询字符串中传递敏感数据，因为查询字符串会被记录，并可在客户端浏览器地址栏中直接看到。

（7）会话管理

由于 Web 应用程序基于无状态的 HTTP 协议生成，因此会话管理是应用程序级任务。检查应用程序的会话管理方法，因为它将直接影响应用程序的整体安整。表 9-8 显示了与会话管理相关的常见漏洞。

表 9-8 常见的会话管理漏洞

漏洞	影响
通过未加密的通道传递会话标识符	攻击者可捕获会话标识符来窃取身份
会话生存期延长	这将增加会话劫持和重播攻击的风险
会话状态存储不安全	攻击者可访问用户的私有会话数据
查询字符串中的会话标识符	可通过在客户端轻松修改会话标识符来窃取身份，然后作为另一用户访问应用程序

测试中需要考虑下列问题，帮助验证应用程序处理敏感数据的方法。

- 如何交换会话标识符。

检查应用程序管理用户会话的会话标识符，以及这些会话标识符的交换方式。考虑下列问题。

① 是否通过未加密的通道传递会话标识符。

如果使用会话标识符（如 cookie 中包含的令牌）跟踪会话状态，检查是否仅通过加密的通道（如 SSL）传递标识符或 cookie。

② 是否加密会话 cookie。

如果使用表单身份验证，确保应用程序使用“<forms>”元素中的 protection="All" 属性加密身份验证。建议同时使用 SSL 和这种方法，以便降低 XSS 攻击的风险，XSS 攻击可设法窃取用户的身份验证 cookie。

③ 是否在查询字符串中传递会话标识符。

确保应用程序不在查询字符串中传递会话标识符。这些字符串可在客户端轻易修改，使用户能作为另一用户访问应用程序，访问其他用户的私有数据，并可能提升特权。

- 是否限制会话生存期。

检查应用程序认为会话标识符有效的时间。应用程序应限制这段时间的长度，以降低会话劫持和重播攻击的威胁。

- 如何确保会话状态存储的安全。

检查应用程序存储会话状态的方法。会话状态可存储在 Web 应用程序进程、ASP.NET 会话状态服务，或 SQL Server 状态存储中。如果使用远程状态存储，请确保 Web 服务器到该远程存储的链接使用 IPSec 或 SSL 加密，以保护在网络中传输的数据。

（8）加密

如果应用程序使用加密来提供安全性，检查加密的内容以及加密的使用方法。表 9-9 显示了与加密有关的常见漏洞。

表 9-9　常见的加密漏洞

漏　洞	影　响
使用自定义加密	安全性总是低于经过测试和考验的由系统平台提供的加密
使用错误的算法，或密钥过小	使用较新的算法可提高安全性，使用较大的密钥也可提高安全性
不能确保加密密钥的安全性	加密数据的安全性与加密密钥的安全性同等重要
在延长的时间段使用同一密钥	如果长时间使用，静态密钥很容易被发现

测试中需要考虑下列问题，帮助验证应用程序处理敏感数据的方法。

- 为何使用特定的算法。

加密只有在正确使用时才能提供真正的安全保障。不同作业使用不同的算法。算法的程度也非常重要。考虑下列问题，评价所使用的加密算法。

① 是否开发自己的加密技术。

不应开发自己的加密技术。众所周知，加密算法和例程的开发非常难，而且很难成功。自定义实施的安全保护一般很弱，基本上不如久经考验的系统平台服务提供的安全措施。

② 是否使用合适的密钥大小来应用正确的算法。

检查应用程序使用的算法及使用该算法的目的。较大的密钥可提供较高的安全性，但会影响性能。对于在数据存储中长时间保存的永久数据，较强的加密非常重要。

- 如何确保加密密钥的安全性。



加密数据的安全与密钥的安全同等重要。要破解加密数据，攻击者必须能检索出密钥和密码文本。因此，需要检查设计，确保加密密钥和加密数据的安全。考虑下列评价问题。

① 如何确保加密密钥的安全。

如果使用 DPAPI，将由系统平台为用户管理密钥。其他情况下，则由应用程序负责密钥管理。检查应用程序确保加密密钥安全的方法。一种较好的方法是，使用 DPAPI 加密其他加密形式所需的加密密钥。然后，安全地保存加密密钥，例如，将其放在配置了受限 ACL 的注册表项目下。

② 回收密钥的频率如何。

不能滥用密钥。同一密钥使用的时间越长，被发现的可能性就越高。设计是否考虑了怎样回收密钥、回收的频率，以及如何将它们分发并安置在服务器中。

（9）参数操作

检查应用程序使用参数的方法。这些参数包括了在客户端和服务器间传递的表单字段、查询字符串、cookie、HTTP 头和视图状态。如果使用像查询字符串这样的参数传递敏感数据（如会话标识符），恶意客户端可轻松使用简单的参数操作逃避服务器端检查。表 9-10 显示了常见的参数操作漏洞。

表 9-10 常见的参数操作漏洞

漏　洞	影　响
无法验证所有输入参数	应用程序易受拒绝服务攻击和代码注入攻击，包括 SQL 注入和 XSS
未经加密的 cookie 中有敏感数据	Cookie 数据可在客户端更改，也可通过网络传递获取并更改
查询字符串和表单字段中存在敏感数据	可在客户端轻松地更改
信任 HTTP 头信息	可在客户端轻松地更改
视图状态未保护	可在客户端轻松地更改

测试中需要考虑下列问题，以帮助确保您的设计不受参数操作攻击影响。

- 是否验证所有的输入参数。

确保应用程序验证所有的输入参数，包括正常和隐藏的表单字段、查询字符串和 cookie。

- 是否在参数中传递敏感数据。

如果应用程序在参数（如查询字符串或表单字段）中传递敏感数据，应检查应用程序使用这种方法而不是更安全的方法（传递会话标识符）的原因。例如，在加密的 cookie 中传递会话标识符。使用这些信息将会话与在服务器状态存储中维护的用户状态相关联。

考虑下列评价问题。

① 是否加密包含敏感数据的 cookie。

如果应用程序使用包含敏感数据的 cookie，如用户名或角色列表，确保它是经过加密的。

② 是否在查询字符串或表单字段中传递敏感数据。

不能这样做，因为就操作查询字符串或表单字段中的数据而言，没有简便的方法可用。实际操作过程中，应考虑使用加密的会话标识符，然后将敏感数据保存在服务器的会话状态存储中。

③ 是否保护视图状态。

如果 Web 页或控件使用视图状态在 HTTP 请求之间维持状态，确保视图状态经过加密，并使用消息验证代码（MAC）检查其完整性。用户可在计算机级配置该设置，也可按页配置。

- 是否为了安全问题而使用 HTTP 头数据。

确保 Web 应用程序不根据 HTTP 头中的信息制定安全决策，因为攻击者可轻松地操作头数据。不要依赖 HTTP 引用站点字段的值来检查源于页的请求是否由 Web 应用程序生成，这将带来漏洞。这种操作本身很不安全，因为引用站点字段可在客户端轻松更改。

（10）异常管理

检查应用程序处理错误的方法。应前后一致地使用结构化的异常处理。同样，确保应用程序不在发生异常时公开太多信息。表 9-11 显示了两大异常管理漏洞。

表 9-11　常见的异常管理漏洞

漏　　洞	影　　响
无法使用结构化的异常处理	应用程序更易受到拒绝服务的攻击，并容易出现逻辑错误，从而暴露安全漏洞
向客户端公开太多信息	攻击者可使用这些信息来规划和调整日后的攻击

测试时应考虑下列问题，以确保设计不易受到异常管理安全漏洞的影响。

- 是否使用结构化的异常处理。

检查应用程序如何使用结构化的异常处理。设计应强制在整个应用程序中使用一致的结构化异常处理。这将创建更强大的应用程序，使应用程序不易处在暴露安全漏洞的不一致状态下。

- 是否向客户端公开了太多的信息。

确保恶意用户无法利用错误信息中的细节信息，考虑下列问题。

① 是否在服务器中捕获、处理和日志记录异常。



确保应用程序不会将内部异常情况传播到应用程序边界以外。异常应在服务器中捕获并记录日志。如果必要，应向客户端返回常规错误信息。

② 是否使用集中的异常管理系统。

在应用程序中一致处理并日志记录异常的最佳方法是，使用正式的异常处理系统。还可将该系统与操作组监视系统性能的监控系统相结合。

③ 是否定义了一组自定义错误信息。

设计必须明确，应用程序在发生严重错误时使用自定义的错误信息。确保这些消息中不包含任何可能被恶意用户利用的敏感数据。

（11）审核和日志记录

检查应用程序的审核和日志记录方法。除了防止抵赖之外，定期分析日志文件有助于识别入侵迹象。表 9-12 显示了常见的审核和日志记录漏洞。

表 9-12 常见的审核和日志记录漏洞

漏 洞	影 响
无法审核失败的登录	入侵企图得不到检测
无法确保审核文件的安全	攻击者可掩饰自己的攻击
无法跨应用程序层进行审核	增加了抵赖的风险

测试中需要考虑下列问题，帮助验证应用程序审核和日志记录的方法。

- 是否明确了要审核的关键活动。

设计必须定义要审核的活动。考虑下列问题：

① 是否审核失败的登录尝试。

这允许用户检测入侵和密码破解企图。

② 是否审核其他关键操作。

确保审核其他关键事件，包括数据检索、网络通信和管理功能（如启用和禁用日志记录）。

- 是否考虑过如何流动原始调用者身份。

设计必须确保跨多个应用程序层来进行审核活动。为此，原始调用者的身份必须在每个层都可用。

① 是否跨应用程序层进行审核。

检验每个层是否都按预期计划对活动进行审核。

② 如何同步多个日志。

日志文件是证明个人犯罪行为和解决抵赖问题的法律程序所必需的。通常，在访问资源的时候，如果由访问资源的同一例程生成审核，则审核最具权威性。确认应用程序

设计中与日志文件同步相关的问题，然后记录某种形式的请求标识符，确保多个日志文件条目可互相关联，并能关联至同一请求。

③ 如何流动原始调用者身份。

如果不在操作系统级流动原始调用者身份（例如，由于此方法伸缩性有限），应明确应用程序如何流动原始调用者身份。对于跨层审核，这是必需的（对于授权来说，可能同样必需）。

此外，如果多个用户映射到同一应用程序角色，应确保应用程序记录原始调用者的身份。

- 是否考虑过保护日志文件管理策略。

检查应用程序设计是否考虑到日志文件的备份、存档和分析。日志文件必须定期存档来确保不被充满；如果充满，应开始回收。而且，还要经常分析日志文件来检测入侵迹象。此外，确保执行备份的账户都是特权最少的，确保仅为备份而公开的所有附加信道安全。

2．应用及传输安全

Web应用系统的安全性从使用的角度可分为应用级的安全与传输级的安全，安全性测试也可从这两个方面入手。

应用级的安全测试的主要目的是查找 Web 应用系统自身程序设计中存在的安全隐患，主要测试区域如下。

- 注册与登录：现在的Web应用系统基本采用先注册，后登录的方式。因此，必须测试有效和无效的用户名和密码，要注意到是否存在大小写敏感，可以试多少次的限制，是否可以不登录而直接浏览某个页面等。
- 在线超时：Web 应用系统是否有超时的限制，也就是说，用户登录后在一定时间内（例如15分钟）没有点击任何页面，是否需要重新登录才能正常使用。
- 操作留痕：为了保证Web应用系统的安全性，日志文件是至关重要的。需要测试相关信息是否写进了日志文件，是否可追踪。
- 备份与恢复：为了防范系统的意外崩溃造成的数据丢失，备份与恢复手段是一个Web系统的必备的功能。备份与恢复根据Web系统对安全性的要求可以采用多种手段，如数据库增量备份、数据库完全备份、系统完全备份等。出于更高的安全性要求，某些实时系统经常会采用双机热备或多机热备。除了对于这些备份与恢复方式进行验证测试以外，还要评估这种备份与恢复方式是否满足Web系统的安全性需求。

传输级的安全测试是考虑到Web系统的传输的特殊性，重点测试数据经客户端传送到服务器端可能存在的安全漏洞，以及服务器防范非法访问的能力。一般测试项目包括



以下几个方面。

- HTTPS 和 SSL 测试：默认的情况下，安全 HTTP（Secure HTTP）通过安全套接字 SSL（Secure Socket Layer）协议在端口 443 上使用普通的 HTTP。HTTPS 使用的公共密钥的加密长度决定的 HTTPS 的安全级别，但从某种意义上来说，安全性的保证是以损失性能为代价的。除了还要测试加密是否正确，检查信息的完整性和确认 HTTPS 的安全级别外，还要注意在此安全级别下，其性能是否达到要求。
- 服务器端的脚本漏洞检验：存在于服务器端的脚本常常构成安全漏洞，这些漏洞又常常被黑客利用。所以，还要测试没有经过授权，就不能在服务器端放置和编辑脚本的问题。这可以通过设计一些相应的测试案例来进行验证。
- 防火墙测试：防火墙是一种主要用于防护非法访问的路由器，在 Web 系统中是很常用的一种安全系统。防火墙测试是一个很大很专业的课题，但这里所涉及的只是对防火墙的功能、设置进行测试，以判断是否满足本 Web 系统的安全需求。

第10章 网络测试

本章首先介绍了网络测试的基本知识，阐述了网络测试的内容和方法，并列举了一些网络测试的指标，使读者能够对网络测试的过程有一定的了解；另外，介绍了网络仿真技术及其在网络测试中的运用，最后探讨了网络应用测试。

10.1 网络测试概述

10.1.1 网络测试发展

自从网络通信产品的诞生起，网络测试技术就成为通信工业中不可或缺的部分。伴随着通信产品的更新换代和网络构建技术的发展，网络测试技术也经历了几个阶段的发展，其技术主体已经逐渐趋于成熟。

20 世纪 1990 年以前，网络产品较少，并且网络架构比较简单，网络测试仅限于验证网络设备和网络的功能，即现在人们常说的功能测试。可以说 20 世纪 90 年代以前基本上是网络产品和数据测试仪表的“史前年代”。

1990 年以后，随着 HUB 等产品的运用，网络产品发展速度加快，世界上先后出现了第一台功能强大的性能测试仪表，解码分析仪表和一致性测试软件/仪表等一系列重要的测试产品。1990 年至 2003 年这段时间，是网络测试技术发展的黄金时期，期间，关于网络设备、网络性能和网络应用的测试技术基本形成比较完整的体系。

国内的网络测试真正得到重视始于 1998 年，一批有远见的数据通信厂商和网络测试技术研究人员开始吸收和引进国际上先进的测试方法和测试设备，并培养出一批网络测试人员。但国内的网络测试发展的时间并不长，技术水平距国际先进水平尚有一定的差距。

目前，国际的测试市场发展到了一个新的阶段，其发展的走势正面临着一些比较大的变革。下面我们探讨其中几个比较值得注意的动向。

1．网络测试的对象从网络层向应用层过渡

类似于丢包率、延迟等指标的测试现在已经做得很纯熟了，虽然这样的测试很重要，但我们需要想一下这个问题：测试的最终目的是什么？答案其实很简单，就是要确保网络能够承载各种各样的应用。最终用户可能不会关心某种条件下设备的丢包率，他可能更在意诸如“能否开展 VOD 业务，能有多少个用户同时上线”等问题。回答这些问题



的最好的办法是对网络上加载不同应用的情况进行测试。从某种意义上说，测试应用才是网络测试的真正意义所在。目前我们看到很多应用测试已经如火如荼地开展起来了，比如话音业务，网站业务等。这将成为一个长久的过程，不断发展下去。

2. 测试重点将逐渐转向可靠性测试

很多用户持有这样的观点：国外的网络产品比国内的更稳定，如果价格允许，要优先考虑采用国外的网络产品。我们不能去指责用户的观点，但有个问题需要我们思考，为什么我们国家的很多产品性能和功能已经相当完备了，用户们还是不大敢用？对用户来说，网络可靠性测试的重要性甚至超过性能测试。网络可靠性的提高同样需要网络测试来促进，而不能仅仅寄希望于设计和开发。目前国内对这方面的测试方法研究还比较少，但是可以预见，网络可靠性测试将成为网络测试技术发展的主要趋势之一。

3. 网络的安全性测试将得到重视

2003 年，几次重大的网络病毒事件为人们敲响了警钟。以前，网络的安全性主要是从终端的安全做起的，然后是防火墙，现在要集成进路由器了。这是个很好的趋势，只有网络中间的中转设备（至少要在各网络的入口设备上）具备安全能力，安全问题才有可能得到解决。安全功能的转移，给测试工作带来很多新的课题，如安全和性能之间如何平衡等。这个发展趋势是必然的，也是很有挑战性的，目前还有待深入地研究。

10.1.2 网络测试意义

网络是信息系统信息共享、信息传递的基础。建立高效、稳定、安全、可靠、互操作强、可预测、可控的网络是网络研究的最终目标，而网络测试是获得网络行为第一手指标参数的有效手段。

随着用户对网络依赖程度的增加，网络的正常运行变得越来越重要，用户对网络可用性、稳定性、响应性（运行效率）等提出了越来越高的要求。网络应用系统越多，网络的功能系统越复杂，出现问题所带来的损失也就越大，网络性能的问题最终会妨碍企业生产效率的提高，并影响到客户服务。

导致网络应用性能降低的因素是多方面的，而网络测试正是一种可以有效提高网络系统及网络应用运行质量的方法，在测量和测试的基础上，建立网络行为模型，并用模拟仿真的方法建立理论到实际的桥梁，是理解网络行为的有效途径。

10.1.3 网络全生命周期测试策略

网络建设生命周期可分为三个阶段，即网络规划设计阶段、网络实施阶段以及网络与应用系统集成阶段。其中规划设计阶段主要的测试目的是利用网络仿真技术测试设计方案，以及对网络设备进行评估测试，为网络选型提供依据；网络实施阶段的主要测试

目的是保证系统可用性和稳定性；网络与应用系统集成阶段的主要测试目的是为了保证应用系统在网络平台上的性能。

网络测试贯穿着整个网络建设生命周期。

1．规划设计阶段

对于复杂的网络系统，设计新网络、增加新网段、对已有网段进行修改、网络服务或网络应用升级等都需要规划设计，提出需求说明书、验证新产品、设计网络拓扑结构等都是规划设计的内容。在此阶段，一方面可以采用网络仿真的方法，评估网络系统规划设计是否合理、是否满足应用系统的运行需求，从而给业主和设计人员提供合理的规划建议。另一方面通过测试评估选择网络设备，优化网络配置，并在模拟实际网络运行的情况下进行性能、稳定性、互操作性测试。通过上述测试，有利于使网络规划设计人员更深入地了解产品设备性能和设备间的互操作性；检验网络能否满足设计所需的全部功能；还有利于预测购买设备等的预算以及进行商务谈判。

2．网络实施阶段

进行网络质量测试，包括吞吐量、包转发率、丢包率等。在网络实施阶段，必须考虑网络的可用性，即网络是否连通、各项网络性能是否达到规划的设想和设计质量要求等。同时，实施阶段的网络质量测试结果也为今后网络应用系统开发提供了网络性能基准值。在最终系统开发完成后，如果网络应用程序的使用效果不是很理想，实施阶段测试得到的网络基准，将有助于定位故障是出现在应用系统上还是网络环境中。

3．网络与应用系统集成阶段

此阶段主要是指应用系统的开发，在开发完成且经过一定的功能和性能测试，证实运行基本稳定后，将程序应用到网络上进行网络级测试，考察多用户并发访问性能、系统响应时间、应用对网络资源的占用情况等。

10.2　网络仿真技术

10.2.1　网络仿真技术概述

网络仿真技术有时也称为网络模拟技术或者网络预测技术。

近年来，随着网络结构日趋复杂，网络规模日趋庞大，新的网络技术层出不穷，网络的应用也越来越多样化。一方面，人们开发出各种新的协议和算法，以满足如服务质量、组播传输、安全、移动网络、策略管理等新的需求。设计和评估这些协议和算法涉及许多问题，尽管实验室中的小规模评价和大范围地在实验床等真实环境下进行实际试验都很有价值，但是每一种都有很大的局限性，缺乏灵活性、扩展性，并且成本太高。

另一方面，由于网络的快速增长而导致的扩展性和异构性，对现在的协议机制和相关的设计方法提出了挑战。在这种情况下，网络仿真作为一种新的网络研究和网络规划设计技术应运而生，无论对于网络协议的设计研究，还是网络规划设计，网络的仿真研究日益显示出其重要意义。

网络仿真技术是一种通过建立网络设备、链路和协议模型，并模拟网络流量的传输，从而获取网络设计或优化所需要的网络性能数据的仿真技术。通常，网络仿真都是在计算机中构造虚拟的环境来反映现实的网络环境，通过数学方法或者动态蒙特卡罗方法来模拟现实中的网络行为，从而有效地提高网络规划和设计的可靠性和准确性，明显地降低网络投资风险，减少投资浪费。

10.2.2 网络仿真的技术原理

网络仿真是一种利用数学建模和统计分析的方法模拟网络行为，从而获取特定的网络特性参数的技术。数学建模包括网络建模（网络设备、通信链路等）和流量建模两个部分。模拟网络行为是指模拟网络流量在实际网络中传输和交换的过程。网络模拟获取的网络特性参数包括网络全局性能统计量、网络节点的性能统计量、网络链路的流量和延迟等，由此既可以获取某些业务层的统计数据，也可以得到协议内部的某些特殊的参数的统计结果。

建模中最基本的概念就是同等性。建模的过程实际上是将实际的系统映射到仿真环境中，仿真环境对实际系统的逼真程度，直接影响仿真结果的有效性。但是由于建模本身的复杂性，仿真系统只能从某些方面去模拟实际系统的行为。因此这里的同等性并不是指仿真系统和实际系统的等同，而是指某些方面、某些层次反映实际的系统。具体的同等性依赖建模人员的定义。在通信网络和分布式系统中包含了从底层通信硬件到高层决策软件涉及的多种技术，为了对系统的性能和行为进行有效的预测，一个成功的系统建模要能够充分反映这些子系统及子系统的交互。由于子系统及其每一层都有很大差别，因此对系统建模一般采用多层次建模的方法，再建立建模域与实际系统的对应关系。

网络仿真采用基于包的建模机制来模拟实际物理网络中包的流动，包括在网络设备间的流动和网络设备内部的处理过程；模拟实际网络协议中的组包和拆包的过程，可以生成、编辑任何标准的或定义的包格式，利用调试功能，还可以在模拟过程中察看任何特定包的包头（Header）和净荷（Payload）等内容。

10.2.3 网络仿真技术应用

1. 网络仿真在网络规划设计中的应用

- 网络仿真能够为网络的规划、设计提供可靠的定量依据。

网络仿真技术能够迅速地建立起现有网络的模型，并能够方便地修改模型并进行仿真，这使得网络仿真非常适用于预测网络的性能，回答“what…if…”这样的问题。例如：“如果网络扩容，骨干中继链路带宽需要扩大多少”“如果网络上增设新的业务，对网络性能有什么影响，网络上的哪些链路或网络设备需要升级和改造”“如果网络拟采用新的技术升级，网络的性能会有多大幅度的改善；这种改善与投入相比是否值得；同时新技术的引进是否会带来负面影响”。

- 网络仿真能够验证实际方案或比较多个不同的设计方案。

在网络规划设计过程中经常出现多个不同的设计方案，它们往往各有优缺点，很难作出正确的选择。因此如何进行科学的比较和取舍往往是网络设计者们感到头疼的事。网络仿真能够通过为不同的设计方案建立模型，进行模拟，获取定量的网络性能预测数据，为方案的验证和比较提供可靠的依据。这里所指的设计方案可以是网络拓扑结构、路由设计、业务配置以及流量规划等。

2．网络仿真在企业网络管理和优化中的应用

- 大中型企业的网络具有复杂的网络结构和协议配置，网络仿真工具可以辅助企业的网络管理人员管理其网络，而且网络仿真往往具有很好的开放性和互联性，可以和当前很多流行的网络管理和监控软件协同进行工作。
- 企业业务通过网路进行传输，应用对网络的可靠性以及有效性具有较强的依赖性，在企业业务达不到服务质量的要求时，如网上交易、数据库等业务响应时间不满足性能需求，仿真器可以捕捉重要的流量进行分析，从业务、网络、服务器等方面找出瓶颈。
- 仿真器通过对整个业务层、流量的模拟，能够有效地查出业务配置中产生的错误，例如某服务器配置不好，易被黑客攻击以及某些业务的参数配置不合适等情况。
- 网络模拟系统可以模拟实际网络中的各种故障，如链路中断，服务器重启等情况。因此对于可能出现的故障，通过模拟，可以知道在这种情况下用户上网会受到怎样的影响，路由协议需要多长时间收敛，路由跳数是否会增加太多，备用链路的带宽是否足够等。
- 通常新增业务需要选择适当的网络做测试，这样成本很高而且具有一定的风险。网络模拟集成了非常多最新的技术和应用（如 MPLS、MULTICAST 等），因此可以模拟开展新业务和增加新用户时的网络状况，分析可能对网络造成的影响。
- 网络模拟的流量模型包含了业务量的信息，因此它可以用来预测不同业务的增长规律和在网络中所占的比重，这对网络规划设计和维护都有重要的参考价值。



3. 网络仿真在网络研发中的应用

- 对于从事新协议研究的研发机构，如军方和政府的研发机构、大专院校等，网络仿真可以有效地模拟协议的各种行为细节，构建接近真实有代表性的网络环境和业务，使得测试结果能够公正地评判新协议的性能。
- 对于大型通信设备制造商，网络仿真工具被作为其网络设备、协议以及应用开发的工具。

4. 仿真的网络预测功能

考虑到电子商务系统、电子政务系统等未来发展的扩展性，预测网络流量的变化、网络结构的变化，对用户系统的影响非常重要。根据规划数据进行预测并及时提供网络性能预测数据。我们利用网络仿真工具可以作到：设置服务水平、完成日常网络容量规划、离线测试网络、网络失效和容量极限分析，完成日常故障诊断、预测网络设备迁移和网络设备升级对整个网络的影响。

通过从网络管理软件获取网络拓扑结构，从现有的流量监控软件获取流量信息（若没有这类软件可人工生成流量数据），可以得到现有网络的基本结构。在基本结构的基础上可根据网络结构的变化、网络流量的变化生成报告和图表，说明这些变化是如何影响网络性能的。网络仿真工具可以提供如下功能：根据预测的结果帮助用户及时升级网络，避免因关键设备超过利用阀值导致系统性能下降；判断哪个网络设备需要升级，这样可减少网络延迟、避免网络瓶颈；根据预测的结果避免盲目的网络升级。

10.2.4 网络仿真软件

1. OPNET 网络仿真软件

（1）OPNET 介绍

OPNET 公司起源于麻省理工学院（MIT），1987 年该公司发布了其第一个商业化的网络性能仿真软件，提供了具有重要意义的网络性能优化工具，使得具有预测性的网络性能管理和仿真成为可能。经过多年的发展，OPNET 公司由于其出众的技术，成为当前业界领先的智能化网络仿真、分析、管理解决方案的提供商，OPNET 产品也被大型通信设备制造商、大中型企业、电信运营商、军方和政府方的研发机构、大专院校等广泛采用，应用于协议及应用开发、网络设备开发、网络规划设计、网络管理和故障诊断等领域。

OPNET 的产品包括 Modeler、ITGuru、SPGuru、WDMGuru 以及 OPNET Development kit。

- Modeler 主要面向设计和研究通信网络、设备、协议和应用，支持所有网络类型和技术。Modeler 为开发人员提供了建模、仿真以及分析的集成环境，大大减轻

了编程以及数据分析的工作量。面向对象的建模方法和图形化的编辑器反映了实际网络和网络组件的结构，因此实际的系统可以直观地映射到模型中。

- IT Guru 是 OPNET 公司开发的一个核心网络仿真软件包。专门为网络专业技术人员和管理人员进行网络规划、设计、建设以及运营提供决策支持。IT Guru 是一个功能齐全、性能优良的网络预测及分析工具软件包。它的主要作用是快速预测网络上的任何变化（如增加新的用户、向新的网络技术转移、推出新的网络应用等），对网络服务等级（如性能）等的影响，指出可能存在的瓶颈之处并提出解决的多种方案。
- SP Guru 面向电信运营等的智能化网络管理、规划以及优化，是一个能够辨识整个网络的独特软件产品，包括网络中的路由器、交换机、协议、服务器以及各种应用业务。内嵌于 SP Guru 的智能代表了当今最新最先进的网络故障诊断、操作验证、规划以及网络设计技术。
- WDM Guru 是一个先进的网络规划解决方案，使得业务提供商和网络设备制造商设计出健壮的且节约成本的光纤网络，并为测试产品提供了一个虚拟的光纤网络环境。WDM Guru 的多层网络架构，大量的技术支持以及当前最新的优化和设计功能使其成为网络设计人员和规划人员的得力助手。
- OPNET 附加功能模块，包括流分析模块（FlowAnalysis）、网络医生模块（NetDoctor）、多提供商导入模块（Multi-Vendor Import）、MPLS 模块等。
- OPNET 开发包（ODK，OPNET Development Kit）和 NetBizODK 是一个更底层的开发平台，其中 ODK 为开发时环境，NetBiz 为运行时环境，可以用于设计用户自定制的解决方案，定制用户的界面，并且 ODK 提供了大量的函数，用于网络规划和优化。

从核心技术来说，OPNET 的全线产品都是基于 Modeler 的核心技术演化和发展而来的，每个产品针对不同的用户群作出了一些特殊的调整和修改。从功能上来说，ODK 的功能最强大，它不仅包含了 Modeler 的建模功能，还包含了界面开发和网络设计的库函数。ITGuru 从功能上看是 Modeler 的一个子集，简单地说，是不具备编程功能的 Modeler。而 SPGuru 也是 ITGuru 的一个超集，它具备了 ITGuru 的所有功能，并且在协议的支持上比 ITGuru 更加全面。

（2）OPNET 技术特点

OPNET 能够准确地分析复杂网络的性能和行为，在网络模型中的任意位置都可以插入标准的或用户指定的探头，以采集数据和进行统计。通过探头得到的仿真输出可以以图形化显示、数字方式观察或者输出到第三方的软件包中去。此外，一系列仿真运行的结果被自动整理到一个单一的 OPNET 输出文件中，以便于比较分析（比如相对于网

络负载的端对端延迟）。

OPNET 模型分为 Network、Node 和 Process 三个层次，Network 模型是最高层次的模型，由网络节点（Node）和连接网络节点的通信链路（Link）组成，由该层模型可直接建立起仿真网络的拓扑结构。Node 模型由协议 Module 和连接 Module 的各种连接组成，如物理接口 Module、MAC Module、IP Module、Route Module、TCP Module、Application Module、Packet Stream、Statistic Wires 等。每个 Module 对应一个或多个 Process 模型，Process 模型由有限状态机来描述，有限状态机用 C 语言编写。用户可以在上述三个层次的任何地方切入编程，建立所需的 Network、Node 或 Process 模型。

OPNET 由厂家提供的标准库模块有：x.25、ATM、FDDI、Frame Relay、Ethernet（10M、100M、1000M）、Token Ring、TCP/IP、UDP、RIP、OSPF、LAPB、TP4、DQDB、HSSB、J1850、STB、CATV、SNA、AMPS、VSAT、Circuit Switching、Client-Sever 等。第三方提供的库模块有地形仿真库、大气仿真库、SUN 网管接口、HP 网管接口等。

OPNET 允许用户使用 FSM（有限状态机）开发自己的协议，并提供了丰富的 C 语言库函数。OPNET 还提供 EMA（外部模块访问）接口，方便用户进行二次开发。

OPNET 支持面向对象的调试。对网络拓扑、节点/设备的体系结构、过程逻辑（状态机）、传输等不同层次的、不同类型的模型，都有专门的、符合人们习惯的工具来进行编辑和浏览，而不像某些软件那样从上到下全部用框图表示。

网络设备厂家（HP、Cisco、3Com、Xylan 等）提供的模型参数全部基于哈佛测试实验室（Harvard Test Lab）的测试结果。

OPNET 支持 SUN、HP、IBM、SGI 工作站和一般 PC 等硬件设备，可以运行在 UNIX、Windows NT 等操作系统平台上。

（3）OPNET 的基本使用

Modeler 进行仿真的一般流程为建立模型、运行仿真到收集最后结果。如图 10-1 所示：

- 理解系统。这一步十分重要，如果不能正确地理解要模拟的系统对象，就无从建模。精确地理解系统成为整个建模的第一个环节，使用者对系统理解的精确性直接影响到所建模型的精确性。
- 理解仿真目的。这里体现了在运行仿真后，仿真的结果能帮助使用者解决什么问题，例如一些常见的问题：如果将以太网 Hub 换成 FDDI 的 Hub，吞吐量会提高多少？如果用户数增加一倍，对业务的响应时间会有什么影响？
- 选择需要建模的方面。从前面的问题中得到建模的目标，如：求某个接收机的吞吐量；测量修改某条链路对重传率的影响；确定系统工作在什么样的负载下开始不稳定。

- 定义输入和输出。输入可能是固定的（如网络的拓扑结构），也可能是变量（如业务产生源的业务产生率）。研究一个系统的时候，一般是保持一些变量不变，然后在一定范围内变化一两个变量，接着就是确定输出内容（如端到端的时延，吞吐量等）以及显示这些输出最好的方法（图形、表、动画等）。
- 确定系统模型。不同的仿真软件可以从不同的地方来描述使用者的系统模型。做好这一步，需要了解使用者的仿真软件能提供的特性，了解如何使用这些特性来描述自己的系统模型。
- 确定输入，运行仿真。大多数变量保持不变，而只是变动其中的一个或两个变量。一般来说，变动的范围是可以事先知道的。
- 系统结果是否精确。结果的容错性和精确性都需要进行验证。一般来说，需要对输出做出一些预测，然后对预测的结果和实际的仿真结果进行比较。
- 结果是否足够详细。根据需要，要么适当地增大输入的范围，要么将输入限制在一个较小的范围。
- 结果是否稳定。如果仿真结果仍然在增加或者减少，并没有达到一个稳定的状态，需要重新运行仿真，使得仿真能够达到稳定状态。如果一个模型不能够达到稳定的状态，也就说明仿真系统本身不是很稳定。

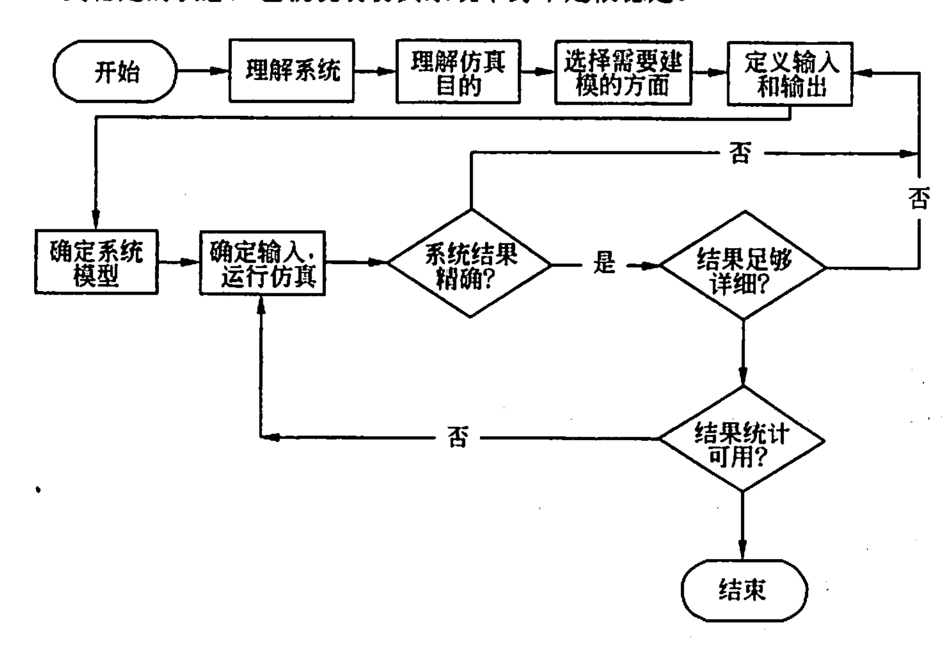


图 10-1　OPNET 仿真流程



2. NS网络仿真软件

（1）NS介绍

NS起源于Real网络模拟器，最初是为了研究分组交换数据网络中的流量控制和拥塞控制方案的动态性。它提供给用户一种方法来描述网络并观察它们的行为，使用C语言编写，并提供源代码便于用户可以根据他们自己的目的修改模拟器。在此基础上，它是由劳伦斯伯克利国家实验室（LBL: Lawrence Berkeley Labs）、美国施乐公司的帕洛阿尔托研究中心（Xerox PARC: Palo Alto Research Center）、加州大学伯克利分校（UCB: University of California, Berkelay）和南加州大学/科学情报研究所（USC/ISI: University of Southern California / Institute for Scientific Information）等合作，由DARPA资助的VINT工程研究得出的一个仿真工具。

NS基于事件驱动模型，支持协议库，广泛采用了开放的体系结构，用户很容易根据自己的需要开发新协议。目前NS支持的协议基本包括了TCP/IP协议域的所有协议：TCP的各种版本、UDP、RTP、Multicast、无线、移动等。NS的另外一个显著特点是允许将实际网络流量引入到网络仿真环境，这样在某种程度上起到了类似测试床的作用，从而可以在一个接近真实的环境中测试协议的性能。NS是一个面向对象的仿真器，由编译和解释两个层次组成：编译层次包括C++类库，解释层次包括对应的Otcl类。用户以Otcl解释器作为前台使用NS。NS内大部分类是TclObject的子类，用户在解释器环境创建新仿真对象，然后镜像到对应的编译层次对象。这样，在不影响效率的前提下，通过Otcl解释器来使用NS提供极大的灵活性和方便性。

（2）NS的技术特点

NS的核心部分是一个离散事件模拟引擎。NS中有一个“调度器”（Scheduler）类，负责记录当前的时间，调度网络事件队列中的事件，并提供函数产生新事件，指定事件发生的时间。有了这个离散事件模拟引擎，原则上用户可以对任何系统进行模拟，而不限于通信网络系统，用户可以自己完成对所要研究的系统的建模工作，编写各种事件的处理代码，然后利用这个离散事件模拟器来完成对这个模型的模拟。

针对网络模拟，NS已经预先作了大量的模型化工作，NS对网络系统中一些通用的实体已经进行了建模，例如链路、队列、分组、节点等，并用对象来实现了这些实体的特性和功能，这就是NS的构件库。用户可以充分利用这些已有的对象，进行少量的扩展，组合出所要研究的网络系统的模型，然后进行模拟，从而减轻了进行网络模拟研究的工作量，提高了效率。NS的构件库所支持的网络类型包括广域网、局域网、移动通信网、卫星通信网等，所支持的路由方式包括层次路由、动态路由、多播路由等。提供了跟踪和监测的对象，也可以把网络系统中的状态和事件记录下来以便分析。还有数学方面的支持，包括随机数产生、随机变量、积分等。

NS 构件库是用两种面向对象的语言编写的：C++和 Otcl。其中 Otcl 是 MIT 开发的 ObjectTcl，即 Tcl 的面向对象的扩展（Tcl 的全称是 Toolkit command language，是一种灵活的、交互式的脚本语言，Otcl 则是在 Tcl 中加入了类、实例、继承等面向对象的概念）。NS 中的构架通常作为一个 C++类来实现，同时，有一个 Otcl 类与之相对应。这种方式被称为分裂对象模型，构件的主要功能在 C++中实现，Otcl 中的类主要提供 C++对象面向用户的接口。用户通过编写 Otcl 脚本来对这些对象进行配置、组合，描述模拟过程，调用 NS 完成模拟。这种方式提高了执行的性能和模拟的效率，增强了构件库的可扩展性和可组合性。

（3）NS 网络仿真的过程

NS 进行网络仿真的方法和一般过程如下：

① 进行模拟前，首先要分析模拟涉及哪个层次。NS 仿真分两个层次：一个是基于 Otcl 编程的层次，利用 NS 已有的网络元素实现模拟，无需对 NS 本身进行任何修改，只要编写 Otcl 脚本；另一个层次是基于 C++和 Otcl 编程的层次，如果 NS 中没有所需要的网络元素，就需要首先对 NS 扩展，添加所需要的网络元素。这需要利用分裂对象模型，添加新的 C++类和 Otcl 类，然后再编写 Otcl 脚本。如图 10-2 所示为 NS 仿真流程。

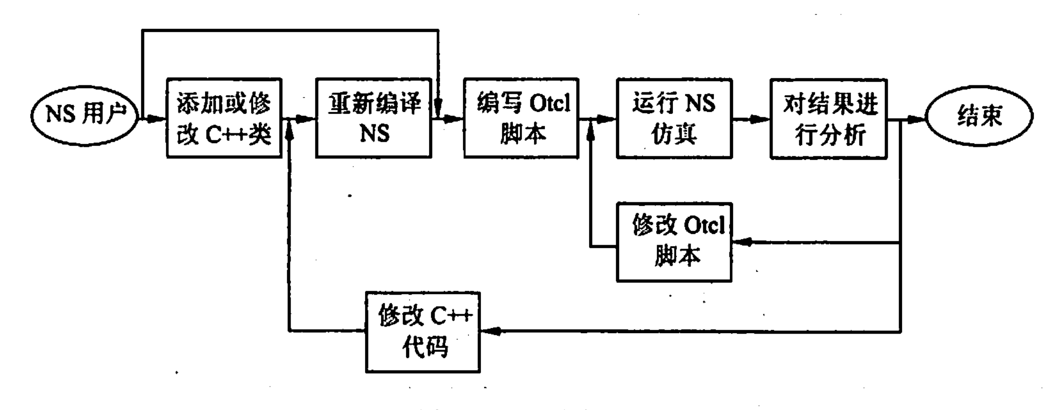


图 10-2　NS 仿真流程

② 开始编写 Otcl 脚本。首先配置模拟网络拓扑结构，此时可以确定链路的基本特性，如延迟、带宽和丢失策略等。

③ 建立协议代理，包括端设备的协议绑定和通信业务量的模型的建立。

④ 配置业务量模型的参数，从而确定网络上的业务流量分布。

⑤ 设置 Trace 对象。Trace 对象能够把模拟过程中发生的特定类型的事件记录在 trace 文件中。NS 通过 trace 文件来保存整个模拟过程。仿真完成后，用户可以对 trace 文件进行分析研究。



⑥ 编写其他的辅助过程，设定模拟结束时间，至此 Otcl 脚本编写完成。

⑦ 用 NS 解释执行刚才编写的 Otcl 脚本。

⑧ 对 trace 文件进行分析，得出有用的数据。也可以用 Nam 等工具观看网络模拟运行的过程。

⑨ 调整配置拓扑结构和业务量模型，重新进行上述模拟过程。

10.3 网络质量测试

10.3.1 OSI 模型简介

开放系统互连参考模型（OSI）是国际标准化组织（ISO）建立的网络模型，具体描述如表 10-1 所示，它建立并规范了一个网络体系中不同组件之间的互连模式，即它可使物理通信方法的设计独立于网络体系中的协议和应用。在网络各设备间的数据传输过程中，每一层都有各自的作用，并为相邻层提供标准接口。

表 10-1 OSI 定义的层模型

序 号	分 层	功 能	样 例
7	应用层	用户接口	Telnet、HTTP
6	表示层	数据如何表示；特殊处理，如加密	ASCII、EBCDIC、JPEG
5	会话层	保持不同应用数据独立	操作系统
4	传输层	可靠或不可靠的传输；重传前的错误纠正	TCP、UDP、SPX
3	网络层	提供路由用来确定路径的逻辑地址	IP、IPX
2	数据链路层	将位合成字节、字节合成帧；用 MAC 地址访问介质；非纠正性错误检测	802.3/802.2、HDLC
1	物理层	在设备间移动位；规定电压、线速和帧输出线缆	EIS/TIA-232、V.35

- 物理层：物理层通过链路来传送比特信息，定义了用户应用程序数据如何转换为 1 和 0 并在物理介质中传递。物理层规定了物理介质的规范，例如电缆和接口规范，也规定了如电压级别、数据速率、最大传输距离等特性。通常一个网络内可以有好几种不同的物理层类型，甚至一个节点也可能有多种不同的物理层类型，这是因为不同的技术要求采用各自的物理层。AUI、10Base-T 和 RJ45 均为该层中的规范。
- 数据链路层：数据链路层（有时也称为链路层）通过物理链路来传输成块的信息。它主要负责处理以下任务：数据出错校验、协调共享媒体的使用（如在一

个 LAN 中）以及编址（当多个系统都可以访问时，如在某个 LAN 中）。另外，不同的链路通常也有不同的数据链路层实现；而且，同一个节点可以支持几种不同的数据链路层协议，节点所连的每一类链路都有自己的协议。以太网、令牌环和帧中继均为数据链路层的实例。

- 网络层：网络层使得网络中的任何一对系统间都可以相互通信。一个全互联的网络是指其中的每一个节点都和其他节点直接相连，但是这种拓扑结构不可能用于有很多节点的情况。比较典型的情况是，网络层必须找到一条通过一系列相连节点的路径，且路径上的每一个节点必须向适当的方向转发数据包。网络层处理的主要任务是：路由计算、数据包的分段和重组（当网络中的不同链路有不同的最大包大小限制时）和拥塞控制。
- 传输层：传输层在两个系统之间建立一条可靠的通信链路。它主要处理一些由网络层引起的错误，比如包丢失和重复包等错误，以及对包进行重新排序、分段（这样运输层用户就可以处理大的报文）和重装（这样可以避免网络层进行低效的分段和重装）。另外，这也有助于传输层在网络发生拥塞时可以相应地降低发送数据的速率。
- 会话层：会话层提供的服务超出了传输层提供的简单全双工可靠通信流，比如对话控制（实现系统间的特殊通信模式）和链接（捆绑一组数据包，使得它们要么都发送，要么都不发送）。如 HTTP 协议可以使用多个 TCP 连接来获取一个网页中所包含的对象，会话层对这些单独的 TCP 连接进行应用协调。
- 表示层：这一层的设计目的是为了对数据的表示取得一致，这样人们就可以定义自己的数据结构，而不必担心比特/字节顺序或者浮点数该如何表示之类的问题了。表示层为应用程序（在应用层中）进行数据格式转换和数据加解密提供了一种普通的方法。表示层具有转换机制，可在 ASCII 文本数据格式与 Unicode 格式间转换，也包括图像文件的压缩技术，如 GIF 和 JPEG。
- 应用层：应用层是 OSI 参考模型的最高层，它为最终用户提供应用环境。应用包括文件传输、虚拟终端及 Web 浏览等，在一个节点上通常有多个应用程序同时运行。

10.3.2　网络测试指标

在对物理层、数据链路层和网络层进行测试时，如以太网，物理层的测试包括碰撞分析、错误统计和是否有随机能量、无格式的帧和信号回波等，数据链路层的测试包括流量分析、错误帧（FCS 错误帧、长帧、短帧和延迟碰撞）统计等，网络层的测试包括响应时间测试、网络层协议分析、IP 路由分析等。



对网络设备和TCP/IP网络的检测主要包括以下基本技术指标。

- 吞吐量（Throughput）。

吞吐量是指被测试设备或被测试系统在不丢包的情况下，能够达到的最大包转发速率。吞吐量测试是在每一对端口上，以全线速度（或测试设置中规定的速率）在测试设置规定的时间段内生成传输流。如果在任何端口上丢失包的话，就将负载减少50%并重新开始测试。然后，用二分搜索法搜索没有包丢失发生时的最大速率。这个速率就是被测试设备的吞吐量，它是按测试设置中规定的每一种包长度测试得出的。

- 丢包率（Packet Loss Rate）。

通过测量由于缺少资源而未转发的包的比例来显示高负载状态下系统的性能。在规定时间内生成100%的负载（或者按测试设置中规定的比例），在测试结束时，报告每对端口应当转发但被丢弃的包的百分比。测试设置中规定的每一种包长度都要进行包丢失测试。

- 延时（Latency）。

延时测试是指测量系统在有负载条件下转发数据包所需的时间。在规定时间内生成100%的负载（或者按测试设置中规定的比例）。在测试过程中，测量每对端口上的每一个包的延时。对于存储转发（Store-and-Forward）设备来说，测量的延时是指从输入帧的最后一个比特达到输入端口的时刻，到输出帧的第一个比特出现在输出端口上的时刻的时间间隔。对于直通式（Cut-Through）设备来说，延时是指从输入帧的第一比特达到输入端口的时刻，到输出帧的第一比特出现在输出端口的时刻的间隔。测试设置中规定的每一种长度的包都要进行延时测试。

- 背靠背性能（Back-to-Back Frame）。

背靠背性能测试是指通过以最大帧速率发送突发传输流，并测量无包丢失时的最大突发（Burst）长度（总包数量）来测试缓冲区容量。在全负载条件下生成突发传输流，如果所有的包都得到转发，就增加突发长度，并重新进行测试。但是，如果某一对端口上出现包丢失，将突发长度减少一半并再次进行测试。然后，利用二分搜索法查找无包丢失时的最大突发长度。测试设置中规定的每一种包长度都要进行背到背性能测试。

另外，还存在其他一些网络质量测试的指标。分别涉及TCP/IP 4～7层负载均衡、IP语音网络测试、宽带xDSL网络性能测试、VPN网络测试、路由测试等测试项，下面作一个简单论述。

（1）TCP/IP 4～7层负载均衡

- 连接建立数：本测试以设定速率执行TCP连接建立。通过改变在多次重复测试中使用的速率，测试系统在一段时间中可处理的输入TCP连接的峰值速率。还可同时测试保持TCP连接的最大数量。

- 会话速率：本测试以设定速率执行 TCP 连接建立与拆除。本测试检测系统在一段时间内可建立和拆除 TCP 连接的峰值速率。
- 连接数据：本测试检测峰值 HTTP 速率。当一个连接建立后，客户机与服务器之间就出现 HTTP 事务处理。这样就可以对基于 URL 的交换技术进行测试，而这项技术是 Web 交换机传送基于 HTTP 请求的传输流的基础。

（2）IP 语音网络测试

测试当语音和数据流经过被测试设备时的语音和数据流的包丢失、单向延时以及延时抖动。此外，得出的参数还被用于为每条语音流生成语音质量评估分析。

（3）宽带 xDSL 网络性能测试

- 帧测试。

① 帧丢失。通过测量由于缺少资源而没有转发的帧的比例，显示给定负载条件下被测试设备的性能。

② 帧延时。测量每端口、每 VPI/VCI 的每个测试帧的延时。

③ 吞吐量。根据 RFC 1242 和 RFC 2544 测量网络吞吐量。

- 端到端 IP 测试。

① 端到端吞吐量。

② 延时。

- 信元测试。

① 信元丢失。测量 ATM（异步传输模式）信元的丢失率。在测试中，帧与信元的长度相等。测试测量传输帧与接收到的帧之间的差。本测试只适用于 ATM 端到端传输。

② 信元延时。利用 ATM 信元特征，测试不同终点之间的延时。

- PPP 容量测试。

① 测量建立 N 个 PPP（点对点协议）会话所需时间。

② 测量建立不同 PPP 会话时的最小、最大和平均延时。

③ 发送 PPP 上的 IP 帧，测试吞吐量和包丢失。

- ATM 完整性。

进行信元错误率（CER）测试，确定通过 ADSL-ATM 网络传送的 ATM 信元传输的准确性。报告收到的 AAL5 CRC 错误帧数量。

（4）VPN 网络测试

- 最大隧道创建数量：VPN 系统能够创建的最大隧道的数量。
- 包丢失：在每条隧道上的数据包丢失。
- 延时：在每条隧道上的包延迟。
- 响应时间：在每条隧道上的应用响应时间。

（5）路由测试

- 支持 BGP4/OSPF/RIP 路由协议。
- 对路由进行压力测试，测试路由的性能。

10.3.3 网络测试类型

根据不同的测试目的和测试对象，网络测试的类型可以概括为以下几类。

① 网络可靠性测试：使被测试网络在较长时间内（通常是 24~72 小时）经受较大负载，通过监视网络中发生的错误和出现的故障，验证在高强度环境中网络系统的存活能力，也就是它的可靠性。可靠性测试作为可接受性测试的一部分，也是比较测试或升级测试的一部分。测试中采用的负载模式很重要，越贴近真实负载模式越好，可靠性测试中使用网络分析仪监控网络运行、捕获网络错误。

② 网络可接受性测试：可接受性测试是在系统正式实施前的“试运行”。它是一个非常有效的方法，确保新系统能提供良好而稳定的性能。可接受测试中也包含多项测试，例如，响应时间、稳定性和特性/功能测试。而在安装或升级网络前，应进行的网络可接受性测试则经常被忽略，事实上可接受性测试能为网络购买者在经济上和技术上提供有力的保证和参考。可接受性测试可以仅在新增加的部件上完成，将已存在的负载加上新增程序或新增组件可能产生负载作为测试使用的负载。

③ 网络瓶颈测试：为找到导致系统性能下降的瓶颈，需要进行网络瓶颈测试。测试中需要测试和计算系统的最大吞吐量，然后再在单个网络组件上进行该项测试，明确各自的最大吞吐量。通过单个组件的最大吞吐量和系统最大可支持吞吐量之间的差额，我们就能发现系统瓶颈的位置以及哪些组件有多余容量。系统瓶颈在不同的测试案例中，出现的位置可能有些变化。例如，一个客户/服务器应用程序测试可能表明服务器是系统的瓶颈，而对一个电子邮件系统的测试，可能表明，广域网连接才是网络的限制因素。如果我们可以在测试的环境中重现引起问题的负载，那么这样的测试结果对我们解决问题有巨大的好处。

④ 网络容量规划测试：进行该测试可检测当前网络中是否存在多余的容量空间，当网络承受的总负载超过网络总容量时，网络的性能或吞吐量就有可能下降，所以在网络负载接近这一临界点（网络的最大容量）前，就要根据负载增长的幅度扩充网络资源。进行该项测试要逐渐增加网络负载，直到网络的运行性能或吞吐量下降至不能达到设计水平的要求为止。网络运行负载和网络最大吞吐量之间的差额就是现有系统的冗余量。

⑤ 网络升级测试：升级测试是将硬件或软件的新版本与当前版本在性能、可靠性和功能等方面进行比较，同时验证产品升级对网络的性能是否会有不良影响。升级测试的关键是要保证被测组件应是运行网络中最关键或最脆弱的组件，该测试还需强调升级

版的新特性，部分新特性测试在升级测试之前作为特征/功能的一部分也可以测试。尽管新产品应该解决了当前版本中的错误，但它们也经常存在一些以前没有出现过的错误，如果这些错误发生在产品的关键部分，将引起严重问题。升级测试不需要测试产品的所有特性，但网络用户正常运行所依靠的关键功能必须在测试之列。

⑥ 网络功能/特性测试：特性测试核实的是单个命令和应用程序功能，通常用较小的负载完成，关注的是用户界面、应用程序的操作以及用户与计算机之间的互操作。特性测试通常由开发人员在他们的开发环境中完成，或是在一个小型网络环境下由测试人员完成。功能测试是面向网络的，核实的是应用程序的多用户特征和重负载下后台功能能否正确地执行，关注的是当多个用户使用应用程序时，网络和文件系统或数据库服务器之间的交互情况。功能测试要求网络的配置和负载非常接近于运行环境下的模式。

⑦ 网络吞吐量测试：吞吐量测试检测的是每秒钟传输数据的字节数和数据报数，用于检测服务器、磁盘子系统、适配卡/驱动连接、网桥、路由器、集线器、交换器和通信连接。吞吐量的测试用于测量网络的性能，找到网络瓶颈以及比较不同产品的性能。吞吐量测试借助某些工具对网络服务器执行文件输入/输出操作来产生流量，或通过某些工具在网上发送专门的数据包或数据帧。

⑧ 网络响应时间测试：检测系统完成一系列任务所需的时间，本项测试是用户最关心的。对于表示层，如微软的 Windows，测试在不同桌面之间切换或装载新负载所需的时间。在不同负载，即不同实际或模拟用户的数目下，运行这一试验，对每个被测试应用程序生成一个负载-响应时间曲线。在应用程序测试中，执行一系列典型网络动作的命令，如打开、读、写、查找和关闭文件，这些命令提供了最好的负载模拟。例如，对每个被测服务器，检测这些命令的响应时间。响应时间测试应该包括对系统可靠性的测试。可靠性问题，如在路由器或服务器中大量丢失数据报文或由于网络组件故障引发的大量坏数据报文，这一切都将严重影响网络的响应时间，因此在整个测试期间都应用网络分析仪监视系统错误。

⑨ 衰减测试：衰减测试是测试贯穿整个通信连接或者信道的信号衰减。必须综合考虑通信连接中所有组件产生的累计衰减，这些组件包括每个插/拔连接件、电源线、UTP 电缆等。在每对连接中都可以测试衰减，测试的方法是在连接的一端发送一定长度的信号（频率大于 100MHz），在连接的另一终端测试信号长度，以确定衰减值。信号辐射、电线阻抗以及绝缘吸收都会引起衰减。总的来讲，信号频率越高，线缆的长度越长，衰减就越大。基本连接的衰减测试可以依据 TSB67 标准来选取衰减值指标。测试信号长度以分贝（dB）为单位，如在频率为 100MHz 时，最大的衰减值不能超过 22dB。

⑩ 网络配置规模测试：利用应用程序响应时间测试和吞吐量测试的测试结果来确定网络组件的规模，还可以利用测试结果和测试者自身对网络体系结构和网络操作的知



识，来调整特定的系统配置组件，改变网络的运行性能。通过反复比较不同的运行性能，并比较每次结果，找到令人满意的运行性能配置。

⑪ 网络设备评估测试：产品评估主要是比较各个产品，例如，服务器、操作系统或应用程序的性能。进行这种测试时，除了待测设备之外，网络中的其他组件都要求保持不变。许多公司的产品评估还包括技术评估和子系统评估。技术评估是指对两种或多种存在竞争的技术在性能方面进行比较，子系统评估是指对包含硬件和软件的网络子系统进行比较。同可接受性测试、可靠性测试一样，评估测试也要进行响应时间、吞吐量和可靠性的测试，这样就能清楚地了解网络中一个组件被另一个组件代替时，对网络产生的影响。

上述 11 项测试类型应根据网络生命周期的各阶段和对网络可能遇到问题的预测，建立一个针对自身主要问题的测试计划。这项测试计划中规定的测试任务在网络生命周期的各阶段应有所不同，并且，当网络扩展或升级时，要改变测试计划以覆盖新的领域，而且测试计划对整个网络中的不同子网的要求应有所不同。要以网络安全和测试成本平衡为原则，提出一个针对需求的合理测试内容表述。测试中，应按照网络测试的需求情况，从这 11 项测试中灵活地选择几项，安排其优先性。以下 3 个测试任务是公认的最重要的测试任务。

① 吞吐量测试：它是标识网络设备、子网和全局网络运行性能的重要指标。

② 可接受性测试：是对将要使用的网络的验收，其重要性和必要性是显然的。

③ 升级测试：运动是永恒的，网络系统永恒的主题是升级换代，升级测试也要不断进行，不要主观地认为升级后的网络一定比原来的好。

10.3.4 网络测试对象

网络测试不可能对整个网路的所有设备和组件进行全部的测试，因此测试内容要有所选择，测试要针对网络系统中的关键部分。可以根据日常监测的有关网络系统的数据，大致划分出网络易出故障的部分；关注新的网络组件（如设备和应用程序等）；采用逐步测试的方式，按一定顺序进行，如：优先进行系统的可接受性测试、性能测试以及升级测试等。

网络测试对象包括 4 种类型。

① 网络平台包括网络操作系统、文件服务器和工作站；

② 应用层是指应用程序的客户端、桌面操作系统和数据库软件等；

③ 子系统主要是指路由器、集线器、交换机和网桥；

④ 全局网路径则是整个网络系统中重要的点对点路径。

网络测试对象还可以进一步细分为 7 个网络子系统，如下几项。

① 文件服务器：这项测试主要是针对服务器的硬件和网络操作系统。容量规划和配置规划测试都要求应用程序和服务器已安装到测试网络中。如果整个网络中的服务器都是标准的，可用同类产品的评估测试代替。

② 工作站：网络性能直接受网络工作站的网卡、协议、缓冲区、视频刷新和桌面管理的影响。工作站的配置规划、吞吐量测试也应和应用程序、操作系统的测试相关联，这样的测试结果更能真实体现运行系统。

③ 网络操作系统：网络操作系统中有很多部分需要测试。但在没有精确评价一个新的操作系统前，只有衰减、稳定性、吞吐量测试得到的测试结果是有效的。在网络组建前期，对于新版操作系统的稳定性测试是非常关键的，它可以保证软件的性能和功能。从测试的角度看，网络操作系统、文件服务器、工作站组成了应用程序和其他网络服务运行的基本平台，它们需要有效的测试。

④ 应用程序、客户/服务器数据库和工作站桌面软件：这些领域是用户直接感觉到的，应用程序最重要的测试任务是稳定性、响应时间、容量、功能和升级测试。对于使用频率高的程序，最关键的测试是升级测试和容量规划测试。桌面和图形用户界面的测试类似于应用程序的测试。总体来说，这些都属于应用层，它们的测试通常一起进行。

⑤ 路由器、集线器、交换机和网桥：从测试的角度看，这类产品的硬件和软件被看作子系统。在设备安装前，要核实设备的稳定性和功能（在运行网络的传输模式下）。这些设备有很多可选的配置方式，而它们在运行网络中采用的配置方式是测试的关键，在考虑工作模式或对网络扩容时，稳定性、功能性和产品评估测试也需要运行。

⑥ 网段：一旦各网络子系统分别进行了测试，就要把它们组合到一起，并使组合后的模拟网段尽可能体现出运行网段的典型特征。完成这项工作需要根据运行网络中网段的不同网络拓扑结构，制定多个网络配置方案。通常把已经在网络中运行的，经确认正常的工作负载模式作为基准，把新的子网加入到作为基准的网段中，然后进行性能、稳定性和功能性测试，看是否有错误发生。

⑦ 全局网：网段测试通过后，就要把网段放到网络中进行全局网测试。连接两个网段的配置，然后在其上重新进行在单个网段中进行过的测试。这项测试对于验证一些网络功能，例如超时、目录可访问性、升级、网络登录和访问权限等是否正确很关键。

10.3.5　网络测试的基本方法

网络测试的方法和手段因测试的目的不同而有所不同。典型的网络设备测试的方法有两种：第一种是将设备放在一个仿真的网络环境中，通过分析该产品在网络中的行为对其进行测试；第二种方法是使用专用的网络测试设备对产品进行测试，如专用的性能分析仪器 SmartBits 6000、IXIA 1600 等。



对于网络系统的布线测试、物理连通性测试以及故障监测也有专门的工具，这些工具是一些底层的网络测试和维护工具，如网络电缆测试仪、令牌环网测试仪、以太网测试仪、光缆测试仪、企业级网络测试仪等，这些都是在网络系统的实施部署和运行维护阶段采用的常用的测试工具。

对于网络协议的一致性测试一般有专门的测试工具来支持，比如说对 ISDN、ATM、ADSL、帧中继等的测试都有专门的测试仪。对网络系统的测试也有相应的测试工具，最典型和最重要的就是网络协议分析仪。网络协议分析仪一般有专用的硬件设备和专门的软件。这类协议分析仪典型的功能是数据包的捕捉、协议的解码、统计分析和数据流量的产生。用协议分析仪我们可以捕捉网上的实际流量、提取流量的特征，据此对网络系统的流量进行模型化和特征化。此外，网络协议分析仪还可以主动地产生大量的数据包施加到网络上，分析网络的响应或对网络系统进行负载测试。目前典型的协议分析仪有 HP 公司的 Internet Advisor（网络专家系统）、WG 公司的 Domino 系列协议分析仪等。另外还有一些纯软件的协议分析工具，有些甚至可以从网上免费下载。但这类协议分析软件无论在协议的解码能力、解码和数据分析的实时性以及数据流量的产生能力上，与用专门硬件实现的协议分析仪相比仍有差距。

还有一些比协议分析仪更高层次的网络性能测试工具，站在应用层的角度使用一些基准业务流量对网络系统的性能进行分析，代表性的软件是 Ganymede Software 公司的 Chariot 软件、Compuware 公司的 Network Vantage 等。

我们在分析和解决网络性能问题时，通常有这么几个分析的模型和方法，如下所示。

- 七层网络结构分析模型法：从网络的七层结构的定义和功能上逐一地进行分析和排查，这是传统的而且最基础的分析和测试方法。这里有自下而上和自上而下的两种思路。自下而上：从物理层的链路开始检测直到应用层。自上而下：从应用协议中捕捉数据包，分析数据包统计和流量统计信息以获得有价值的资料。
- 网络连接结构的分析法：从网络的连接构成来看可以大致分成客户端、网络链路、服务器端三个模块。

① 客户端也具备网络的七层结构，也会出现这样或那样的故障，从硬件到软件，从驱动到应用程序，从设置错误到病毒等。

② 来自网络链路的问题通常需要网络电缆测试仪、网管软件、协议分析仪来帮助确定问题的性质和原因。这个方面的问题分析要有坚实的网络知识和实战经验，在很大程度上实战经验会决定排除故障的时间。

③ 在分析服务器端的情况时，要了解服务器的硬件性能及配置情况，软件系统性能及配置情况，网络应用及对服务器的影响情况。

- 工具分析法：有强大的各种测试工具和软件，它们的自动分析和专家系统能快速地给出网络的各种参数，甚至是故障的分析结果。
- 经验分析法：靠时间、错误与成功的积累，大多数的网络测试工程师都是采用这个方法，再结合网管和测试工具迅速定位网络的故障。

10.3.6　网络测试标准及工具

1．测试标准

在网络测试领域，没有一个处于支配地位的公司或机构成功地建立起测试的标准。IETF（Internet Engineering Task Force）下属的 BMWG（Benchmarking Methodology Working Group）工作组，制定了一系列测量各种全局网络技术性能的建议。今后的工作组会更进一步地对在这些技术上搭建的系统与业务提出建议。每一个提出的建议中都描述了所涉及的设备、系统或业务的类型；讨论与这个类型相关的性能特性；清晰地确定对这些特性加以描述的指标；规定对这些指标进行收集的测试方法；最后提出对测试结果进行报告的要求，通常是通过 RFC 发布标准，而且尽量独立、公正（不受厂家技术指标的影响）。

例如，针对交换机的测试，BMWG 公布了下面这些主要的 RFC。

- RFC 1242：网络设备及网络设备互联基准术语。
- RFC 2544（RFC 1944）：网络设备互联基准测试方法。
- RFC 2285：局域网交换机测试术语。
- RFC 2432：IP 多播基准。
- RFC 2647：网络防火墙性能测试术语。

除此之外，还有一些常用的，但 BMWG 未做标准的测试项。从这个工作组制定的 RFC 及草案中，可以看出他们的一个工作思路，一般先针对被测对象制定一套术语，随后采用这套术语对被测对象定义测试的方法和手段。

为了适合我国网络发展的实际情况，我国开展了测试标准制定工作，在 IP 网络设备标准以及测试标准方面制定了下列标准：

- YD/T1098-2001 路由器测试规范——低端路由器。
- YD/T1141-2001 千兆位以太网交换机测试方法。
- YD/T1142-2001 IP 电话网值守设备技术要求及测试方法。
- YD/T1156-2001 路由器测试规范——高端路由器。
- YD/T1072-2000 IP 电话网关设备测试方法。
- YD/T1075-2000 网络接入服务器（NAS）测试方法。

在国内外测试标准的推动下，测试工作取得了很大进展，但是这些标准多是针对网



络设备的，换句话说，是用于网络设备或网络组件（如网络协议栈）测试，而不是用于多个网络组件有机结合后的网络子网或整个网络测试的。因此在缺乏统一测试标准的情况下，对整个网络或子网的测试就需要参考大量的技术规范，并对测试内容和测试方法等一系列技术规则进行总结归纳。

2．测试工具

（1）物理线缆测试仪

常见的测试项目主要有线缆长度、衰减、阻抗、串扰、反射和噪声等。某些线缆测试仪还可以定位线缆路由，即由线缆测试仪将一系列音频信号输入到线缆中，并用一个小的附属设备（充当音频放大器）在 30～40cm 处监听信号，这样即可探测到地板下或隔板下的线缆路由情况。此外，还可以使用附属信号发生器测试引起的分配情况并检测布线故障（如线缆折断、短路或线对反转等）。在使用线缆测试仪时，必须让其工作在要求的频率范围内，因为像串扰、衰减等参数都直接与信号频率有关。例如，对高速数据传输技术（如快速以太网或 ATM）来说，线缆测试的频率范围是 1~100MHz，在 TSB67（电信系统公告牌 67，1995 年 9 月）规范中详细描述了线缆测试方法及相应的精度需求，还定义了两个频率精度等级（Ⅰ级和Ⅱ级），其中Ⅱ级测试仪的精度比Ⅰ级测试仪高。任何价格昂贵的线缆测试仪都必须遵照《TSB67Ⅱ级规范》，当然，在某些特殊场合下进行网络故障检测和修复，有 TSB67Ⅰ级线缆测试仪就足够了。有很多优秀的物理线路测试工具，如美国 Agilent 公司的线缆认证测试工具 WireScope 155 和 FLUKE 公司的 DSP-4100 等。

（2）网络运行模拟工具

模拟工具是指按照指定网络基准或网络负载模式，以指定速率向所连网络发送指定大小的数据包，从而模拟出所需的网络流量状况，进而再现运行网络真实的环境。

（3）协议分析仪

协议分析仪是定位和排除故障的关键工具，可以捕获网络上的数据报或数据帧。一个数据包或数据帧主要包含三方面信息：源地址和目的地址、数据、控制位。捕获的数据包存放在磁盘缓冲区中，可以对各种协议进行进一步的解析。解析的程度可以不一样，可以进行简单的报文类型或报文地址解析，也可以进行复杂的解析，对数据部分进行分析，还原为指令代码，如文件打开、关闭等操作。协议分析仪可以监控网络的数据流量、连接数、处在网络连接中的目的端和源客户端的地址（MAC、IP、SPX）、数据包的大小分布、协议分布等，可以通过历史采样功能对网络参数进行采样，并通过直方图或饼图显示。网络维护人员用分析仪捕获数据包，查看数据包，解析数据包，由此获取信息，再分析这些信息，检查网络问题。网络协议分析仪还可以主动地产生大量的数据包施加到网络上，分析网络的响应或对网络系统进行负载测试。协议分析仪有许多不同的测试

模块，最简单的测试系统就是安装在PC机（要配置相应的LAN和WAN接口）上的软件系统，而高性能的协议分析仪，一般都采用专用的硬件设备和基于专家系统的高性能分析软件。究竟选用何种协议分析仪，应取决于待测网络的规模、复杂性和拓扑结构等因素。使用得较多协议分析仪有NAI公司的Sniffer、FLUKE公司的OptiView、HP公司的Internet Advisor（网络专家系统）、WG公司的Domino系列、免费网络协议分析软件Ethereal等。

（4）专用网络测试设备

专用的软硬件结合的测试设备，能够对网络设备、网络子网以及整个网络系统提供综合测试，具有典型的三大功能：数据捕获、负载产生和智能分析。常见的有Spirent公司的SmartBits 6000、IXIA公司的IXIA 1600等。下面简单介绍一下SmartBits，该产品是数据通信领域广泛认同的，能够对网络及设备进行性能测试和评估分析的标准测量仪表，为进行10/100/1000 M以太网、ATM、POS、光纤通道、帧中继网络和网络设备的高端口密度测试提供了行业标准。SmartBits提供了测试xDSL、电缆调制解调器、IPQoS、VoIP、MPLS、IP多播、TCP/IP、IPv6、路由、SAN和VPN的测试应用，可以测试、仿真、分析、开发和验证网络基础设施并查找故障，从网络最初的设计到对最终网络的测试，SmartBits提供了产品生命周期各个阶段的分析解决方案。SmartBits 6000在一个机架中最多可支持96个10/100 Mbps以太网端口、24个千兆以太网端口、6个万兆以太网端口、24个光纤通道端口、24个POS端口或上述端口的任意组合，并可通过使用SmartBits多机扩展功能，将多达512台设备同步连接起来。

（5）网络协议的一致性测试工具

对于网络协议的一致性测试，一般有专门的测试工具来支持，比如说对ISDN、ATM、ADSL、帧中继等的测试都有专门的测试仪。

（6）网络应用分析测试工具

以应用性能分析为主要目的的网络性能测试软件，如Compuware公司的Application Vantage应用产品包，从服务器、网络到客户端。提供强大的故障定位和解决方案，以快速定位和解决问题。

10.4 网络应用测试

10.4.1 网络应用监控

1. 概述

网络监测功能一般是通过将网络探测器设备（如运行探测器软件的PC或专用设备）



安装在网络的某一网段上采集数据实现的。探测器常常被插入到局域网交换机上的镜像端口中，即被配置为复制来自交换机上另一个端口的传输流的端口。探测器将只能够从镜像端口采集传输流数据。

2．网络应用监控目标

在系统试运行之后，需要及时准确地了解网络上正在发生什么事情：什么应用在运行，如何运行；多少 PC 正在访问 LAN 或 WAN；哪些应用程序导致系统瓶颈或资源竞争，定位问题的根源是在客户端、服务器、应用程序还是网络。在大多数情况下，用户较关心的问题还有，哪些应用程序占用大量带宽，哪些用户产生了最大的网络流量等。

3．网络应用监控原理

对于使用集线器连接的网络而言，以太网协议的工作方式使网络内源主机发送的，写有目的主机 IP 地址的数据包，发往同一集线器上的所有主机。但是，这种数据包并不能在协议栈的高层直接发送出去。要发送的数据包必须从 TCP/IP 协议的 IP 层交给网络接口。数字信号到达一台主机的网络接口时，在正常情况下，网络接口读入数据帧后进行检查，如果数据帧中携带的物理地址是自己的，或者物理地址是广播地址，则将数据帧交给上层协议软件，也就是 IP 层软件，否则就将这个帧丢弃。对于每一个到达网络接口的数据帧，都要进行这个过程。换言之，用集线器连接的网络，网内任何一台机器都能够“听到”其他机器的通信，当然也能够将这些通信包抓取下来，这正是网络监控能够实现的前提。所以，在集线器网络中，任何一台机器上如果布署了网络监控软件，则无论数据帧的地址是什么，都不会将帧丢弃，而是将所有的数据帧都交给上层协议软件，对数据包进行分析处理，达到监控的目的。

而对于采用交换机、路由器等网络设备的网络而言，由于交换机与集线器最大的不同是通信数据包不再复制到其他所有端口，而是“精确”地发往目标机器所在的那个端口，所以，其他机器就无法“听到”这种目的性较强的通信，也就无法实现数据包的抓取了。要想在基于交换机的网络中部署网络监控，必须采取一些其他手段，在网络的“关口”处设岗，即指所有机器的通信都会流经的端口设置监控。如：通过代理服务器上网的网络需要在代理服务器上部署监控；如果局域网的网关是计算机，则在网关计算机上部署；如果网络的网关是路由器的话，则在交换机和路由器之间加装一个集线器，从而实现监控；其他方法还包括交换机端口镜像等。

4．网络应用监控工具

Network Vantage 是一个全面应用监控和报告产品，它帮助发现和优化网络上的应用性能。Network Vantage 采用成熟的软件探针技术，采用被动监听技术监控网络，Network Vantage 自动发现应用，追踪通过 LAN/WAN 结构的应用流量，收集详细的性能列表数

据。Network Vantage 关联这些信息，并输出到一个用户图形接口——交互式用户界面，它自动识别应用性能是如何在网络上表现的。通过这个交互式用户界面可以找到每个受到影响的工作站、服务器、网段和当日的时间段。这种方式可以确定关键信息如：什么工作站引起最多流量、哪些混合流量（应用和时间段）在关键的 WAN 网络链路上造成过载、应用或服务器的网络响应时间趋势等。Network Vantage 提供的信息有助于分析问题根源，快速决定受影响的服务器数量和用户数。

Sniffer 是另一种网络监控工具，它可以实现如下功能：

- 捕捉网络流量；
- 诊断问题；
- 监控现实情况下的网络活动；
- 针对网络上单独的工作站、会话以及部分网络，搜集详细的使用和错误信息；
- 存储历史的使用和错误信息用于基线分析；
- 产生可见或可视的报警信息；
- 利用工具在网络上打探针，模拟流量、测量响应时间、计算跳转以及定位问题。

10.4.2 网络故障分析

1．网络故障诊断步骤

网络故障以某种症状表现出来，故障症状包括一般性的（如用户不能接入某个服务器）和较特殊的（如路由器不在路由表中）。对每一个症状使用特定的故障诊断工具和方法都能查找出一个或多个故障原因。一般故障诊断模式如下。

- 当分析网络故障时，首先要清楚故障现象。应该详细说明故障的症候和潜在的原因。为此，要确定故障的具体现象，然后确定造成这种故障现象的原因和类型。例如，主机不响应客户请求服务。可能的故障原因是主机配置问题、网卡故障或路由器配置命令丢失等。
- 收集需要的用于帮助隔离可能故障原因的信息。向用户、网络管理员、管理者和其他关键人物提一些和故障有关的问题。广泛地从网络管理系统、协议分析跟踪、路由器诊断命令的输出报告或软件说明书中收集有用的信息。
- 根据收集到的情况考虑可能的故障原因。可以根据有关情况排除某些故障原因。例如，根据某些资料可以排除硬件故障，把注意力放在软件原因上。对于任何机会都应该设法减少可能的故障原因，以便尽快策划出有效的故障诊断计划。
- 根据最后的可能的故障原因，建立一个诊断计划。开始仅用一个最可能的故障原因进行诊断活动，这样可以容易恢复到故障的原始状态。如果一次同时考虑一个以上的故障原因，试图返回故障原始状态就困难多了。



- 执行诊断计划，认真做好每一步测试和观察，直到故障症状消失。
- 每改变一个参数都要确认其结果。分析结果确定问题是否解决，如果没有解决，继续下去，直到解决。

2．软件问题的诊断

软件问题的诊断建立在网络应用分析的基础上。

目前测试需要面对的大多数系统是复杂的分布式多层应用，这些应用的特点是：网络是应用中的一个组件，应用对网络产生影响，同时网络对应用也会产生影响，这样的特点造成应用在网络上的容量需求很难估计，失败的风险较大。如何针对分布式多层应用进行网络故障定位，以及在应用线程级分析中的应用是迫切需要解决的问题。网络应用分析可以发现多种问题，例如，客户端是否对数据库服务器运行了不必要的请求；当服务器从客户端接受了一个查询，应用服务器是否花费了不可接受的时间与数据库服务器通信等。借助于网络应用分析工具，可以在投产前调整应用在网络上的性能。

目前成熟的网络应用分析工具很少，并且从事网络故障分析的成本很高。为了实现分布式应用分析，规避应用实施风险，可以重点考虑以下几个方面的测试：

① 优化应用程序的性能。

② 预测响应时间。

③ 确定网络带宽需求。

④ 在应用程序领域和网络领域分别进行故障定位。

（1）网络应用分析的关键因素

网络应用分析的关键因素包括：会话信息、包信息、响应时间信息、负载信息、高峰信息、线程信息等。同时还可以采用一些网络应用分析技术，例如，响应时间预测技术与带宽模拟技术等。

- 会话信息。指应用程序节点之间的会话信息，会话信息主要包括会话往返行程和会话流量信息。

① 往返行程。一个往返行程是一对节点之间的一系列帧请求/回应。如图 10-3 所示的会话中，往返行程个数是 2。一个应用程序如果具有较少的往返行程次数，那么它受网络品质的影响，例如网络延迟的影响较小，反之则较大。

② 流量信息。会话的流量信息包括节点之间传输的字节数或者帧数目，如图 10-4 所示表示的是节点之间传输的字节数。

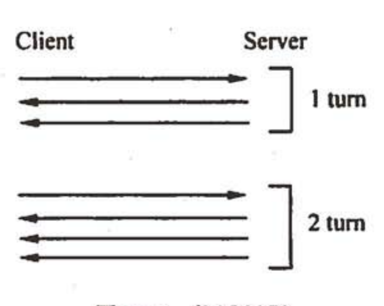


图 10-3 往返行程

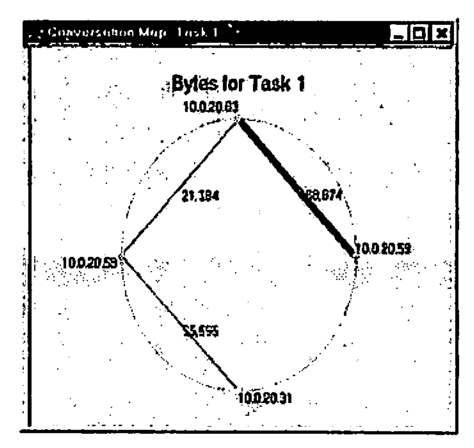


图 10-4 节点之间传输的字节数

当捕捉到的节点很多，并且流量纷繁交织时，可以设定某种条件过滤流量。如图 10-5 和如图 10-6 所示，表示的是过滤前和过滤后的流量显示状态。

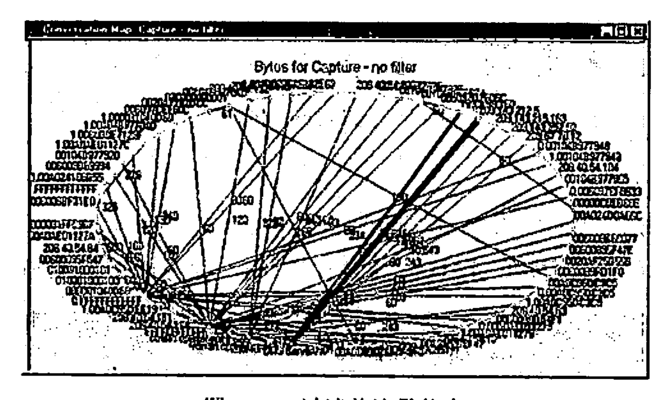


图 10-5 过滤前流量状态

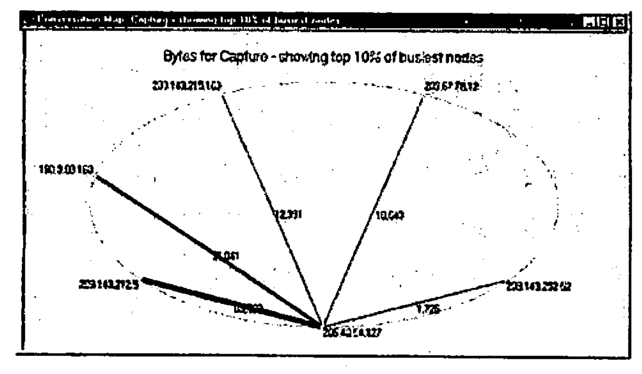


图 10-6 过滤后流量状态

- 包信息。

可以先对包信息进行解码，然后分析包的详细信息，还可以分析包与包之间的关系、一个时间段内包的数量和包尺寸平均大小，以及包与线程的关系等。如图 10-7 所示为利用工具监控与应用相关的包大小及传输方向。大量的红色表示不充足的流量，传输线之间的间隔表示存在延迟问题。

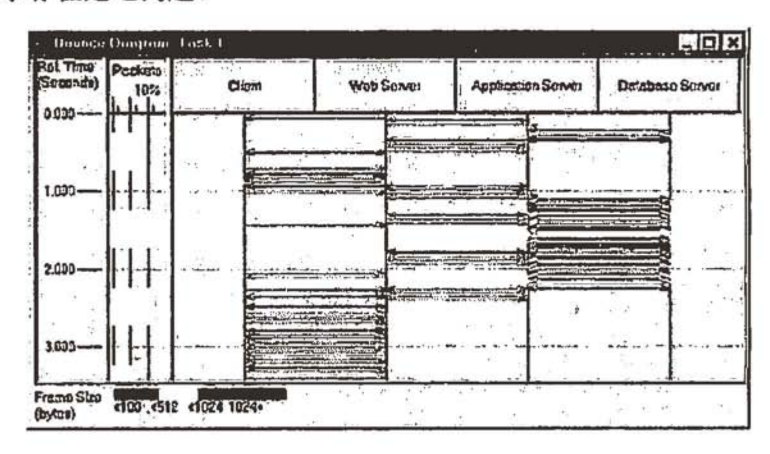


图 10-7 包的大小及传输方向

- 响应时间信息。

如图 10-8 所示，分解出客户端响应时间、网络传输时间以及服务器（包括 Web 服务器、应用服务器、数据库服务器等）的处理时间，为分段定位故障提供依据。

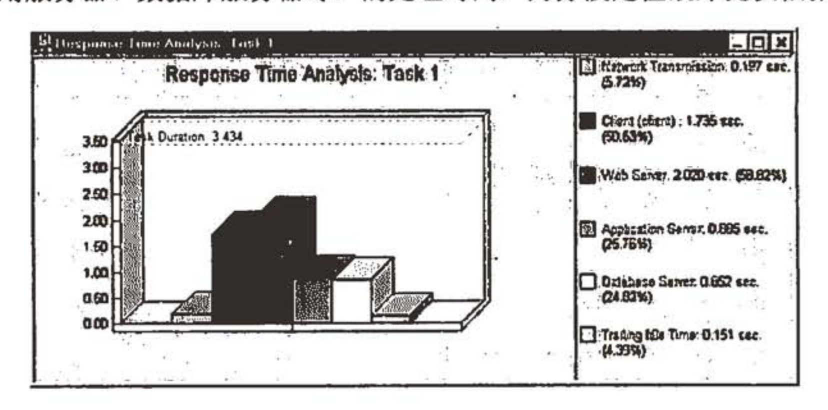


图 10-8 响应时间信息

- 负载信息。

如图 10-9 所示显示了有效负载与其他负载的比例，来评估与业务相关的流量效率。

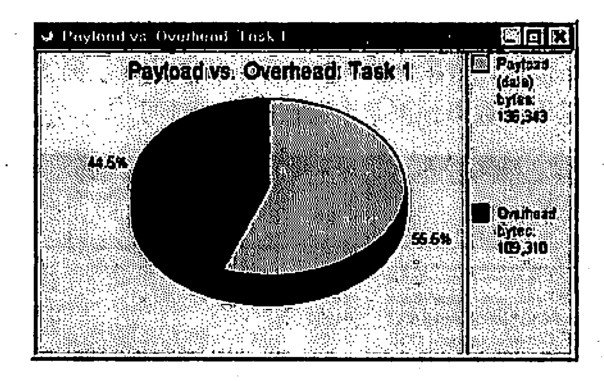


图 10-9　负载信息

- 高峰信息。

需要明确高峰发生的时刻以及高峰期的流量，以确定广域网的容量需求，如图 10-10 所示表示了一个应用的平均流量和高峰流量。

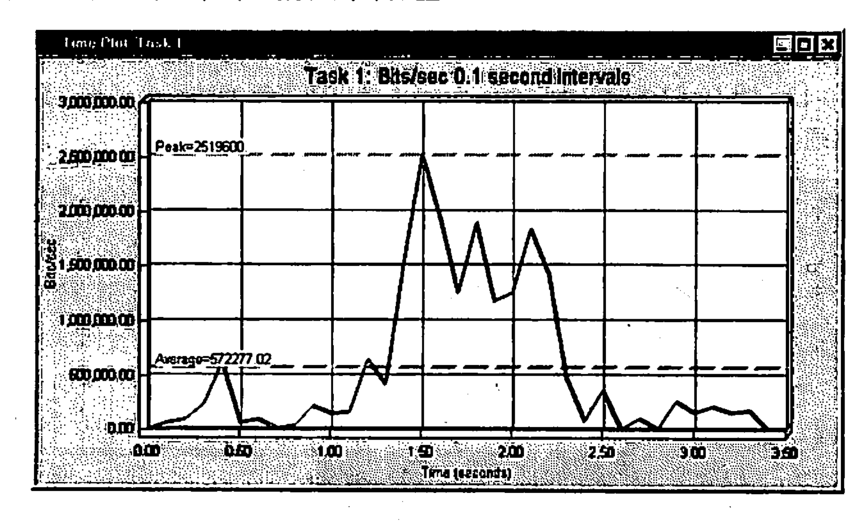


图 10-10　高峰信息

- 线程信息。

可以对某个线程进行分析，也可对线程之间的关系进行分析。如图 10-11 所示，每个线程中都包含了一组包信息，所以可以分析线程和包之间的关联。线程也可以解析为协议信息进行分析。在线程分析时还可以将有关联的线程设置为线程组来分析。

- 响应时间预测。



可以利用工具为实验室中得到的测试数据进行响应时间预测。要完成这项工作需要定义三类参数，分别是带宽、背景负载（利用率）以及延迟，例如可以模拟在与服务器通信的带宽、负载、延迟参数下预测某个会话的响应时间。

	Name	Start Time
1	⊟All threads	0.00000
2	HTTP:GET /secure-frame?buck	0.00000
3	HTTP:GET /secure-frame?buck	0.00300
4	TCP:Seq# 850186054 Ack# 835	0.09900
5	ORACLE:Connect request (153	0.18200
6	ORACLE:Connect request (153	0.26105
7	ORACLE:Miscellaneous comma	0.27500
8	ORACLE:"select PASSWORD f	0.28932
9	ORACLE:"select * from SL_USE	0.30800
10	ORACLE:"select A_BUCKET_C	0.31668
11	ORACLE:"select * from body_hi	0.33237

图 10-11 线程信息

如图 10-12、图 10-13 和图 10-14 所示，可以看出随着带宽的逐渐递增，数据传输时间以及网络通信时间的变化趋势。

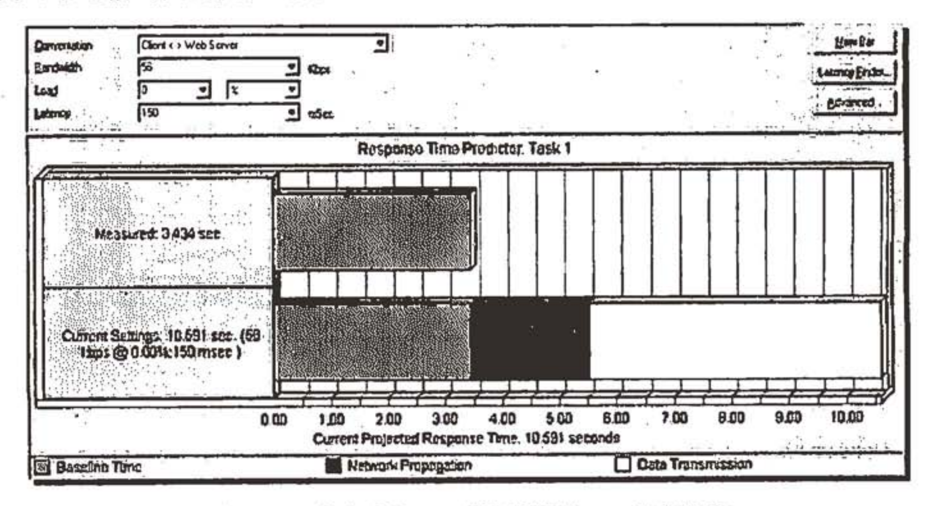


图 10-12 客户端和 Web 服务器通过 56K 连接通信

- 带宽模拟。

带宽模拟的目的是为系统传输选择一个合适的带宽，使得系统的峰值流量可以接受。这些测试数据为选择带宽提供了依据。

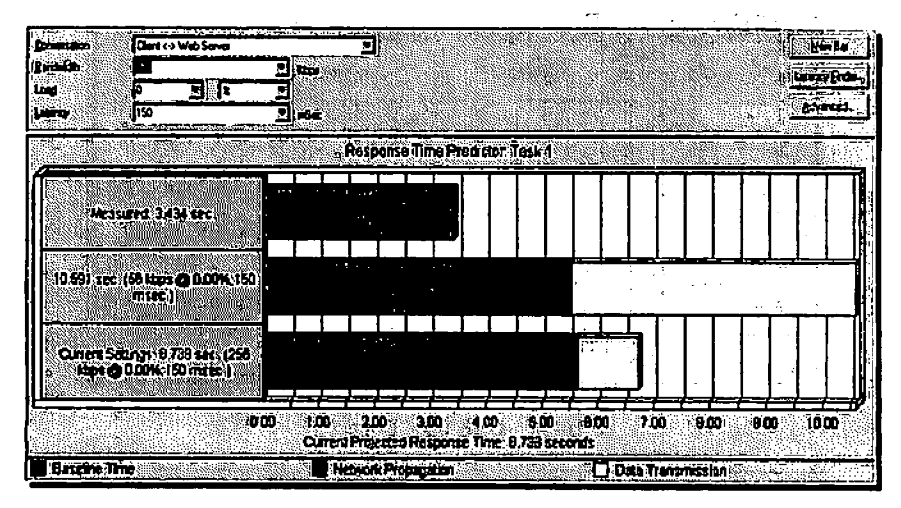


图 10-13　客户端和 Web 服务器通过 256K 连接通信

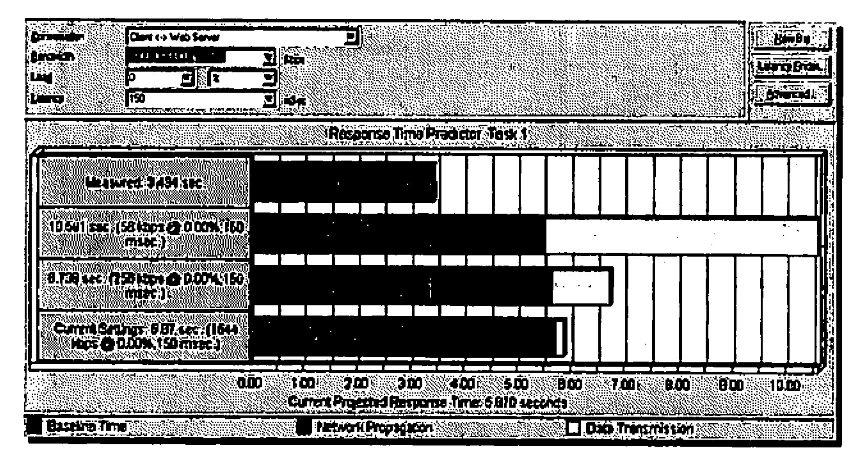


图 10-14　客户端和 Web 服务器通过 T1 连接通信

（2）故障定位举例

下面举两个例子，说明如何对性能问题进行故障定位。

- 第一个例子命名为 Task1，访问一个网页，例如www.cnn.com，速度较慢，问题出在哪里？为了实现故障定位，需要关注的问题如下。

① 性能问题在哪里。

如图 10-15 所示，在初始的 DNS 请求之后有一个 8 秒的间隔，也就是服务器为了响应 DNS 请求花费了 8 秒钟时间。

如图 10-16 和图 10-17 所示，线程 4、5、6 所关联的数据包之间有 10 秒的时间间隔。



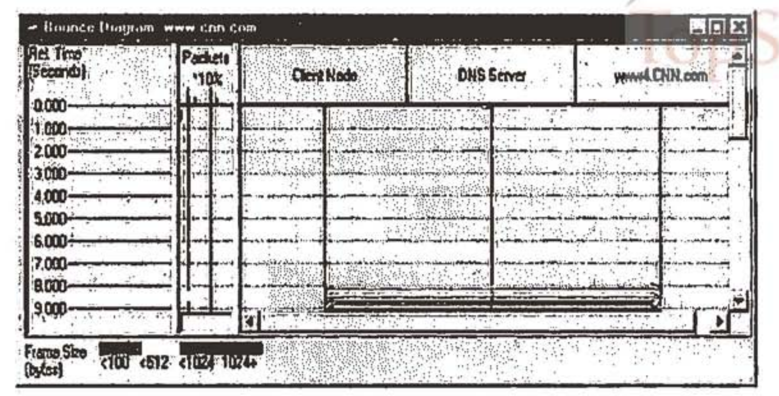


图 10-15 DNS 请求 Bounce 图

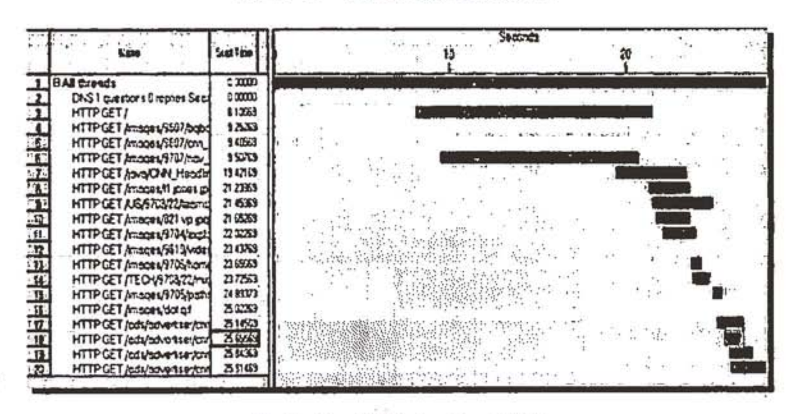


图 10-16 线程 4、5、6 信息

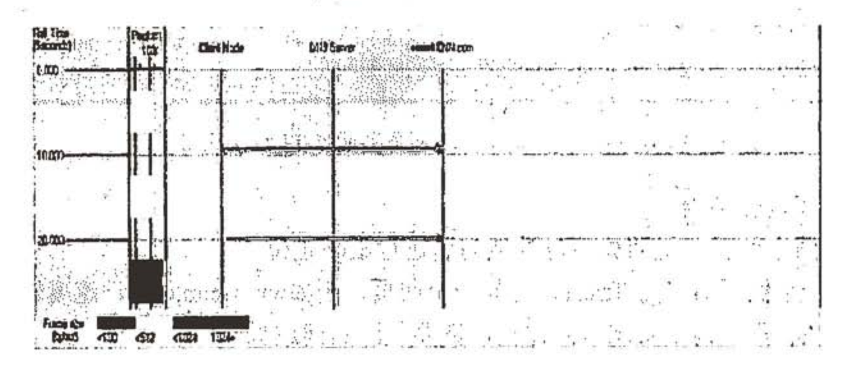


图 10-17 与线程 4、5、6 相关数据包的 Bounce 信息

② 这个应用涉及哪些服务器。

这个应用涉及一个 DNS 服务器和一个 Web 服务器“www..CNN.com”。如图 10-18

所示为应用会话图。

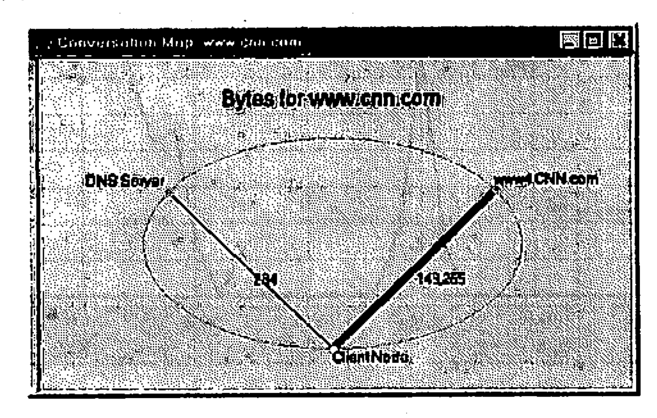


图 10-18　应用会话

③ 流量中哪些是有效的。

从图 10-19 所示可知有效负载达到 90.9%，应用流量效率较高。

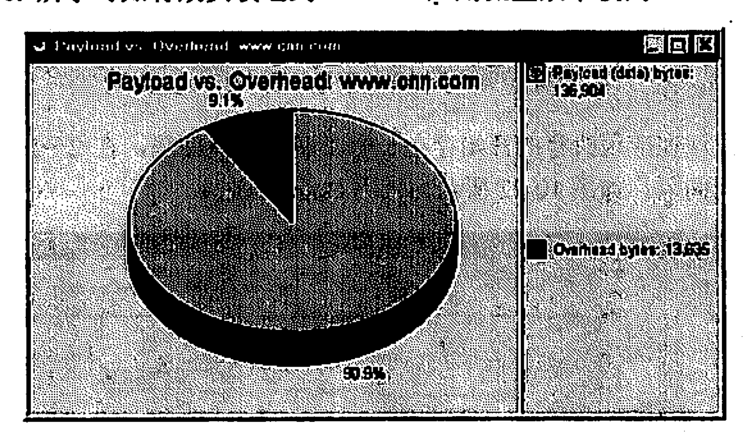


图 10-19　流量负载

④ 流量的高峰期如何。

从图 10-20 可以看到，25～30 秒之间应用产生了持续流量高峰，可以将这个高峰与相关执行的线程关联起来。

- 第二个例子是一个对比测试，我们来看如图 10-21 所示的 Task1 与 Task2 的测试数据。

Task1 的执行时间是 3.43 秒，Task2 的执行时间是 5.71 秒。为什么 Task2 的流量小于 Task1，而执行时间却长于 Task1？是否存在性能问题？



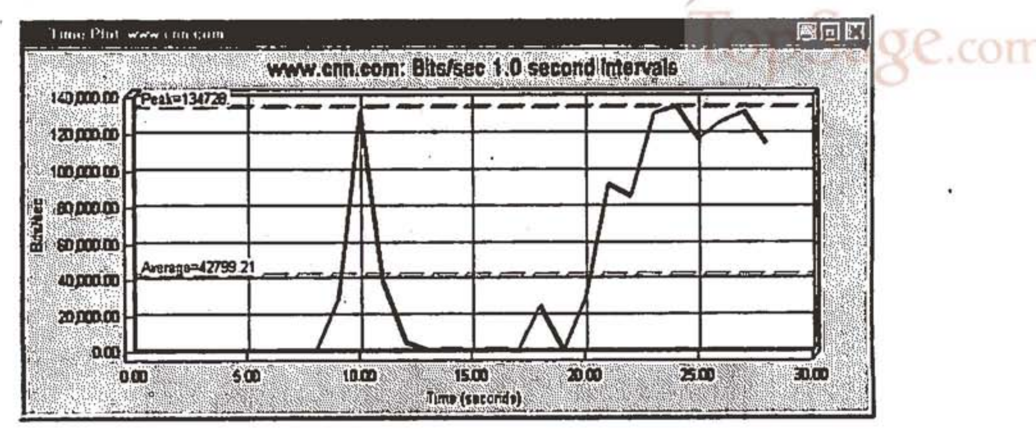


图 10-20 流量高峰期

Task	Duration	Frames	App. Frames	%App.	Bytes	Avg. Frame	App. Turns	TCP Turns	Frames/App	Bytes/App.	Rate (bits/sec)
Task 1	3.43	1,729	1,642	95.02%	245,653	142.18	609	39	2.14	303.65	572,277.02
Task 2	5.71	406	377	92.86%	73,589	181.25	183	12	2.22	402.12	103,062.09

图 10-21 Task1 与 Task2 应用数据

首先我们发现在应用执行的开始存在一个 1 秒的时间间隔，在 2～5 秒之间有一个超过 2 秒的时间间隔。如图 10-22 所示为应用 Bounce 图。

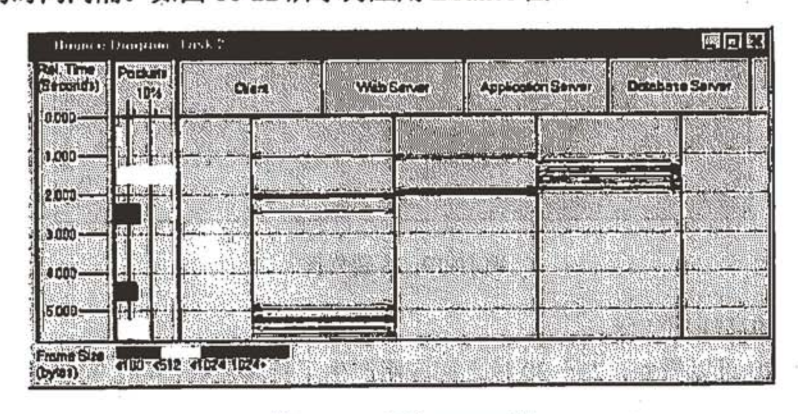


图 10-22 应用 Bounce 图

那么问题发生在客户端、服务器还是在网络上？

客户端发出一个请求，Web 服务器响应，在 1 秒之后又发出了同样的 HTTP/GET 请求，说明第一个请求失败。如图 10-23 所示为 HTTP 线程与 Bounce 关联图。

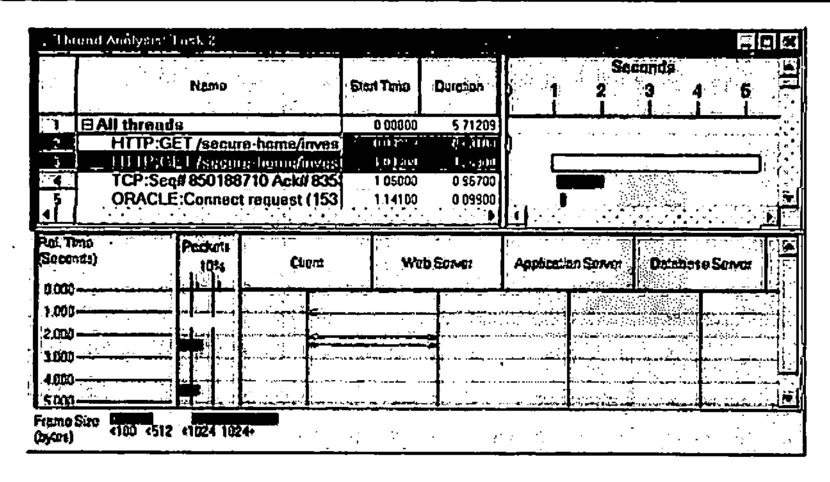


图 10-23　HTTP 线程与 Bounce 关联图

最后一个 Oracle SQL 请求在物理时刻 1.65 秒时发生，而下一个 HTTP/GET 请求直到物理时刻 4.88 秒时才发生，说明在请求之间存在较长时间间隔，服务器空闲。

从如图 10-24 所示的 Oracle 线程与 Bounce 关联图可知，每个时间间隔之后，系统都在等待客户端，可以初步将故障定位在客户端，下面可以进一步验证。

图 10-24　Oracle 线程与 Bounce 关联图

从图 10-25 可以看出，响应时间分析显示客户端响应时间为 4.6 秒，总的响应时间为 5.7 秒，客户端响应时间占总响应时间的 80%左右，客户端的响应时间较长。下面从线程代码级再做进一步验证。

从图 10-26 所示的线程分析可以看到，发生延迟的 HTTP 请求都发生在客户端。



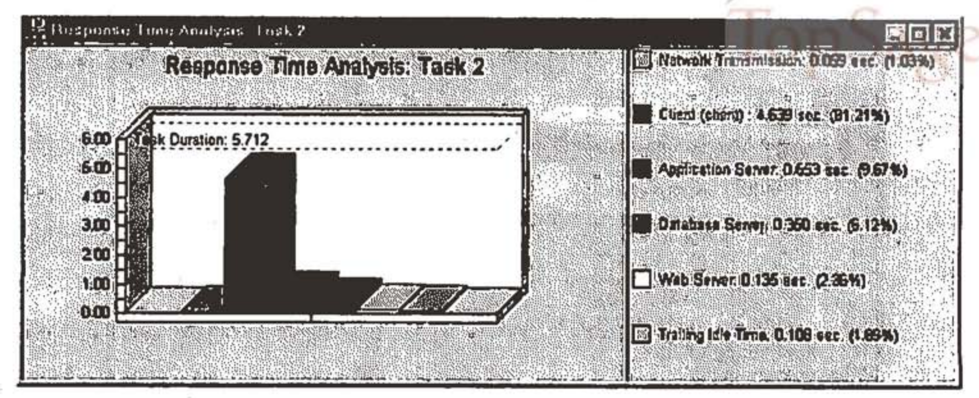


图 10-25 响应时间分析

	Name	Start Time	Duration
1	⊟ All threads	0.00000	5.71209
2	HTTP:GET /secure-home/inves	0.00000	0.00100
3	HTTP:GET /secure-home/inves	1.00300	4.26300
4	TCP:Seq# 850188710 Ack# 835	1.06000	0.96700
5	ORACLE:Connect request (153	1.14100	0.09900
6	ORACLE:Connect request (153	1.24005	0.01108
7	ORACLE:Miscellaneous comma	1.25118	0.02688
8	ORACLE:"select PASSWORD f	1.27811	0.00187
9	ORACLE:"select MAGIC_NUMB	1.28003	0.01530

图 10-26 线程分析

3．硬件问题的诊断

（1）物理层及其诊断

物理层是 OSI 分层结构体系中最基础的一层，它建立在通信媒体的基础上，实现系统和通信媒体的物理接口，为数据链路实体之间进行透明传输，为建立、保持和断开计算机和网络之间的物理连接提供服务。

物理层的故障主要表现在设备的物理连接方式是否恰当、连接电缆是否正确、MODEM 或 CSU/DSU 等设备的配置及操作是否正确等方面。

确定路由器端口物理连接是否完好的最佳方法是，使用 show interface 命令，检查每个端口的状态，解释屏幕输出信息，查看端口状态、协议建立状态和 EIA 状态。

（2）数据链路层及其诊断

数据链路层的主要任务是，使网络层无须了解物理层的特征而获得可靠的传输。数

据链路层为通过链路层的数据进行打包和解包、差错检测，并具有一定的校正能力，协调共享介质。在数据链路层交换数据之前，协议关注的是形成帧和同步设备。

查找和排除数据链路层的故障，需要查看路由器的配置，检查连接端口的共享同一数据链路层的封装情况。每对接口要和与其通信的其他设备有相同的封装。通过查看路由器的配置检查其封装，或者使用 show 命令查看相应接口的封装情况。

（3）网络层及其诊断

网络层提供建立、保持和释放网络层连接的手段，包括路由选择、流量控制、传输确认、中断、差错及故障恢复等。

排除网络层故障的基本方法是：沿着从源到目标的路径，查看路由器路由表，同时检查路由器接口的 IP 地址。如果路由没有在路由表中出现，应该通过检查来确定是否已经输入适当的静态路由、默认路由或者动态路由。然后手工配置一些丢失的路由，或者排除一些动态路由选择过程的故障，包括 RIP 或者 IGRP 路由协议出现的故障。

10.5　结论

现今的网络测试，尤其是网络应用测试与软件测试的关系日渐紧密，通过网络测试可以获得网络行为的指标参数，可以有效地提高网络系统及网络应用运行质量。我国的网络测试还处于起步阶段，需要参考国外测试领域的相关资料和文献，不断发展提高测试技术。



第 11 章　安全测试与评估

11.1　概述

软件系统的安全性是信息安全的一个重要组成部分，而软件是由程序、数据和文档组成的。根据国家标准 GB/T 16260—1996《信息技术　软件产品评价质量特性及其使用指南》，软件安全性是与防止对程序及数据的非授权的故意或意外访问的能力有关　的软件属性。所以，针对程序和数据的安全性测试是软件安全性测试的重要内容。

在软件的质量特性中，安全性与功能、易用性有较大的关联，安全性是通过某些功能的实现来体现的，易用性与安全性紧密联系，同时也存在矛盾。通常易用性强的软件系统安全性相对较差，安全性较高的软件系统易用性也比较弱。

在进行安全性测试时应当考虑软件系统应用领域，以及软件系统本身对安全性的要求，并且安全性需求应该与功能、易用性等需求相协调，取得软件系统整体性能的平衡点。

11.2　测试与评估内容

11.2.1　用户认证机制

用户认证就是指软件系统用户在使用软件或系统时，必须提供用户身份证明，然后软件系统根据用户数据库的资料，开放特定的权限给登录用户。

最普通的用户认证就是口令，口令具有共享秘密的属性。例如，要使服务器操作系统识别要登录系统的用户，最简单的口令认证是用户将他的用户名和口令传输给服务器。服务器就将该用户名和口令与数据库里的用户名和口令进行比较，如果相符，就通过了认证，可以访问系统资源。

目前主要的用户认证机制有如下几种。

- 数字证书。这是一种检验用户身份的电子文件，提供较强的访问控制，并具有较高的安全性和可靠性。这种证书可以授权购买。
- 智能卡。这种解决办法可以持续较长的时间，并且更加灵活，存储信息更多，并具有可供选择的管理方式。

- 双重认证。系统不是采用一种认证方式，而是采用两种或多种认证方式，这些认证方式包括令牌、智能卡和仿生装置，如视网膜或指纹扫描器等，例如同时使用 ATM 卡和 PIN 卡进行双重认证。
- 安全电子交易（SET）协议。它是电子商务中安全电子交易的一个国际标准。其主要目的是解决信用卡电子付款的安全保障性问题：保证信息的机密性，保证信息安全传输，不能被窃听，只有收件人才能得到和解密信息。保证支付信息的完整性，保证传输数据完整地被接收，在中途不被篡改。认证商家和客户，验证公共网络上进行交易活动的商家、持卡人及交易活动的合法性。广泛的互操作性，保证采用的通信协议、信息格式和标准具有公共适应性。从而可在公共互联网络上集成不同厂商的产品。

用户认证机制是保证数据安全的基础，因此有必要对用户认证机制进行全面的测试，评价认证机制的合理性。

11.2.2　加密机制

加密机制是保护数据安全的重要手段，加密的基本过程就是对原来为明文的文件或数据，按某种算法进行处理，使其成为不可读的一段代码，通常称为“密文”，使其只能在输入相应的密钥之后才能显示出明文内容，通过这样的途径来达到保护数据不被非法窃取、阅读的目的。该过程的逆过程为解密，即将该编码信息还原为其原来数据的过程。

密码函数可用来作为加密、解密、保证数据完整性、鉴别交换、口令存储与检验等的一部分，借以达到保密和鉴别的目的当用于机密性的加密时，密码技术被用于把敏感性较强的数据（即受保护的数据）变换成敏感性较弱的形式。当用于保证数据完整性或鉴别交换时，密码技术被用来计算不可伪造的函数。加密开始时，在明文上实施以产生密文，解密的结果或是明文，或是在某种掩护下的密文。使用明文作通用的处理在计算上是可行的，它的语义内容是可以理解的。除了以特定的方式，密文是不能用来计算的，因为它的语义内容已隐藏起来。有时故意让加密是不可逆的（例如截短或数据丢失），这样做的目的是不希望导出原来的明文，例如口令。

密码技术能够提供或有助于提供相关保护，以防止消息流的观察和篡改、通信业务流分析、抵赖、伪造、非授权连接、篡改消息等行为的出现。主要用于密码的保护、数据的传输过程中的安全防护、数据存储过程的安全防护等。

不同加密机制或密码函数的用途、强度是不相同的，一个软件或系统中的加密机制使用得是否合理，强度是否满足当前需求，是需要通过测试来完成的，通常模拟解密是测试的一个重要手段。



11.2.3 安全防护策略

安全防护策略是软件系统对抗攻击的主要手段，安全防护策略主要有安全日志、入侵检测、隔离防护、漏洞扫描等。

安全日志是记录非法用户的登录名称、操作时间及内容等信息，以便于发现问题并提出解决措施。安全日志仅记录相关信息，不对非法行为作出主动反应，因此属于被动防护的策略。

入侵检测系统是一种主动的网络安全防护措施，它从系统内部和各种网络资源中主动采集信息，从中分析可能的网络入侵或攻击。一般来说，入侵检测系统还应对入侵行为作出紧急响应。入侵检测被认为是防火墙之后的第二道安全闸门，在不影响网络性能的情况下能对网络进行监测，从而提供对内部攻击、外部攻击和误操作的实时保护。

漏洞扫描就是对软件系统及网络系统进行与安全相关的检测，以找出安全隐患和可被黑客利用的漏洞，同时漏洞扫描技术也是安全性测试的一项必要手段。

隔离防护是将系统中的安全部分与非安全部分进行隔离的措施，目前采用的技术主要有两种，即隔离网闸和防火墙，隔离网闸属于近两年新兴的网络安全技术，主要目的在于实现内网和外网的物理隔离，防火墙是相对成熟的防护技术，主要用于内网和外网的逻辑隔离。

以上四种安全防护策略通常会结合应用，但是任何防护措施都存在局限性。软件系统的安全性与软硬件设备的安全特性、人为制定的安全防护规则等息息相关。所以安全防护策略、软硬件设备的安全特性，以及人为制定的安全防护规则都在测试的范围内。

11.2.4 数据备份与恢复手段

备份与恢复是一种数据安全策略，从软件系统本身角度，我们认为任何一款软件系统都应当提供数据的备份与恢复功能，对自身的数据进行保护。目前通常的做法是利用数据库原有的备份与恢复机制，这种方式的实现比较简单，通常用于中小型软件系统的备份。对于大型软件系统来说，通常的做法是通过备份软件把数据备份到磁带上，在原始数据丢失或遭到破坏的情况下，利用备份数据把原始数据恢复出来，使系统能够正常工作。理想的备份系统是全方位、多层次的。例如：使用硬件备份来防止硬件故障；如果由于软件故障或人为误操作造成了数据的逻辑损坏，则使用软件方式和手工方式结合的方法恢复系统。这种结合方式构成了对系统的多级防护，不仅能够有效地防止物理损坏，还能够彻底防止逻辑损坏。理想的备份系统成本太高，不易实现。在设计备份方案时，往往只选用简单的硬件备份措施，而将重点放在软件备份措施上，用高性能的备份软件来防止逻辑损坏和物理损坏。

数据备份与恢复技术通常会涉及以下几个方面。

- 存储设备：磁盘阵列、磁带机（磁带库）、光盘库、SAN 设备。
- 存储优化：DAS、NAS、SAN。
- 存储保护：磁盘阵列、双机容错、集群、备份与恢复。
- 存储管理：数据库备份与恢复、文件与卷管理、复制、SAN 管理。

从上面可以看出，数据备份与恢复是涉及计算机工程和软件工程的综合安全技术，因此有必要对软件或系统的数据备份与恢复的效果进行测试，保证实际数据的安全性。

11.2.5　防病毒系统

计算机病毒的发展日益猖獗，已经成为软件系统的大敌，因此有必要采用全面的计算机病毒控制手段进行病毒的防范，基本的防毒技术包含如下几部分。

1．集中式管理、分布式杀毒

对局域网进行远程集中式安全管理，统一升级杀毒引擎和病毒定义，并可通过账号和口令设置移动控制台。并且采用先进的分布技术，利用本地资源和本地杀毒引擎，对本地节点的所有文件全面、及时、高效地查杀病毒，同时保障用户的隐私，减少网络传输的负载，避免因大量传输文件而引起网络拥塞。目前，杀毒软件中较为流行的两种集中管理方式如下。

- 以策略为中心（如图 11-1 所示）。

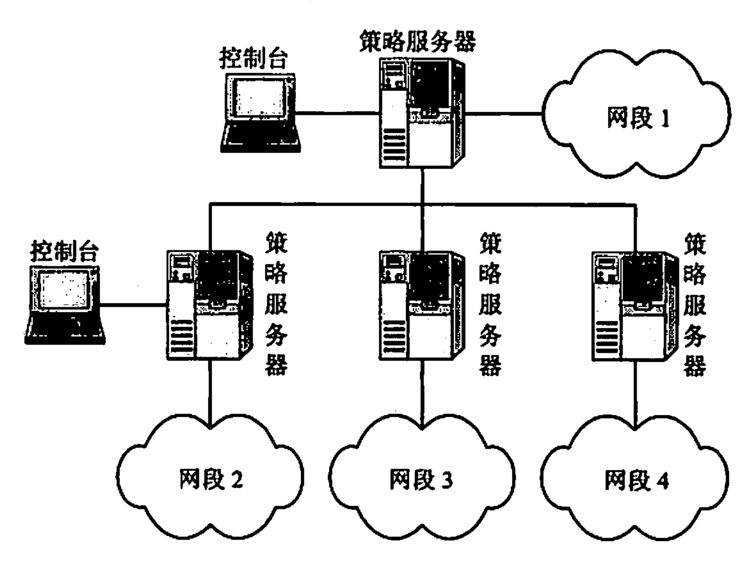


图 11-1　以策略为中心进行网络杀毒管理



以逻辑上的策略域进行杀毒策略的部署，一般由策略服务器（或中心服务器）负责实现网络杀毒策略的部署，这种方式可以脱离网络拓扑结构，部署较灵活。

- 以服务器为中心（如图 11-2 所示）。

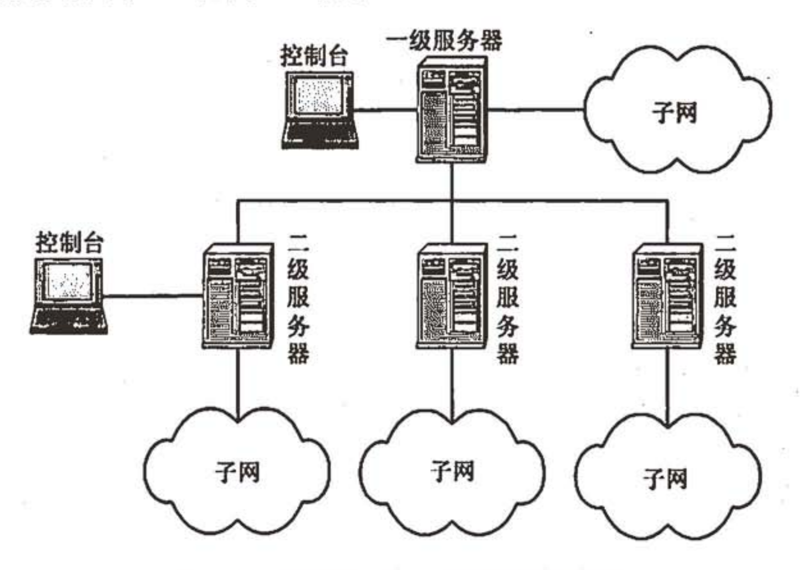


图 11-2　以服务器为中心进行网络杀毒管理

以物理上的网络服务器为中心，进行网络杀毒的管理，这种方式与网络拓扑结构融合，使管理更加方便。

2．数据库技术、LDAP 技术的应用

由于网络杀毒工作的日益复杂，一些网络杀毒软件厂商已经开始使用数据库技术和 LDAP 技术进行策略和日志的存储、用户管理等，不但增强了用户管理能力、策略组织能力、提高了策略调用速度，而且便于以后向日志分析等方面扩展。

3．多引擎支持

我们知道，对于网络安全来说，只有相对安全而没有绝对的安全，而对于杀毒引擎来说，我们并不能说一种杀毒技术或杀毒引擎能够查杀任何病毒，因此对于多引擎技术来说，即使使用多项杀毒技术进行网络杀毒，可以有效地提高网络杀毒的成功率，但是必然会增加网络杀毒软件的复杂度。

4．不同操作系统的防护

由于计算机网络应用的不断增加，导致计算机病毒入侵途径日益增多，并且对于某些网络上的设备，如 Linux、UNIX 服务器来说，虽然其本身受病毒侵害的程度相对较低，但是却也成为病毒存储的温床和发源地。因此，网络杀毒体系将会从单一设备或单一系

统，发展成为一个整体的解决方案，并与网络安全系统有机地融合在一起。对软件系统来讲，无论是服务器还是客户端都应该得到保护。目前，现有的网络杀毒软件通常都可以扩展到对于数据库服务器、文件服务器、邮件服务器、Lotus 服务器等的病毒防护。

5．远程安装或分发安装

由于网络杀毒软件有可能通过远程方式，实现不同物理区域的数十台，甚至成百上千台，客户端服务器的杀毒模块的安装，因此要求管理员本地安装是不现实的。目前系统一般提供两种方式进行客户端的远程安装，一种是通过 Windows 系统远程控制命令进行批量客户端的远程安装，另外一种方式是让所有的用户通过 Web 页面下载客户端自行安装，任何一种方式都需要通过管理手段和技术手段实现。

测试人员在测试的时候应该对网络防毒软件的选购、部署方式、策略定义进行评估和测试。

11.3　安全系统测试策略

11.3.1　基本安全防护系统测试

基本安全防护系统一般采用防火墙、入侵检测、漏洞扫描、安全审计、病毒防治、Web 信息防篡改、物理安全等基础安全技术，以保障应用系统的安全。针对不同的安全技术，设计的测试点如下。

- 防火墙。

① 是否支持交换和路由两种工作模式；

② 是否支持对 HTTP、FTP、SMTP 等服务类型的访问控制；

③ 是否考虑到防火墙的冗余设计；

④ 是否支持对日志的统计分析功能，同时，日志是否可以存储在本地和网络数据库上；

⑤ 对防火墙本身或受保护网段的非法攻击系统，是否提供多种告警方式以及多种级别的告警。

- 入侵监测系统。

① 能否在检测到入侵事件时，自动执行切断服务、记录入侵过程、邮件报警等动作；

② 是否支持攻击特征信息的集中式发布和攻击取证信息的分布式上载；

③ 能否提供多种方式对监视引擎和检测特征的定期更新服务；

④ 内置的网络能否使用状况监控工具和网络监听工具。

- 漏洞扫描。

能否定期或不定期地使用安全性分析软件，对整个内部系统进行安全扫描，及时发现系统的安全漏洞、报警，并提出补救建议。

- 病毒防治。

① 能否支持多种平台的病毒防范；

② 能否支持对服务器的病毒防治；

③ 能否支持对电子邮件附件的病毒防治；

④ 能否提供对病毒特征信息和检测引擎的定期在线更新服务；

⑤ 防病毒范围是否广泛，是否包括 UNIX 系列、Windows 系列、Linux 系列等操作系统。

- 安全审计。

① 能否进行系统数据收集，统一存储，集中进行安全审计；

② 是否支持基于 PKI 的应用审计；

③ 是否支持基于 XML 的审计数据采集协议；

④ 是否提供灵活的自定义审计规则。

- Web 信息防篡改系统。

① 是否支持多种操作系统；

② 是否具有集成发布与监控功能，使系统能够区分合法更新与非法篡改；

③ 是否可以实时发布和备份；

④ 是否具备自动监控、自动恢复、自动报警的能力；

⑤ 是否提供日志管理、扫描策略管理和更新管理。

11.3.2 安全系统防护体系

对业务系统进行全面保障的安全体系，主要体现在以下 7 个层次。

① 实体安全：基础设施的物理安全；

② 平台安全：网络平台、计算机操作系统、基本通用应用平台（服务/数据库等）的安全；

③ 数据安全：系统数据的机密性、完整性、访问控制和可恢复性；

④ 通信安全：系统之间数据通信和会话访问不被非法侵犯；

⑤ 应用安全：业务运行逻辑安全/业务资源的访问控制；业务交往的不可抵赖性/业务实体的身份鉴别/业务数据的真实完整性；

⑥ 运行安全：对系统安全性的动态维护和保障，控制由于时间推移和系统运行导致安全性的变化；

⑦ 管理安全：对相关的人员、技术和操作进行管理，总揽以上各安全要素并进行控制。

安全性测试策略针对这 7 个层次进行测试和评估。

安全系统的主要构成一般包括证书业务服务系统、证书查询验证服务系统、密钥管理系统、密码服务系统、授权管理服务系统、可信时间戳服务系统、网络信任域系统、基本安全防护系统等。在后续的章节中，我们将针对这些关键技术，提出详细的测试策略。

1．证书业务服务系统测试

证书业务服务主要包括证书认证及证书审核注册两项功能。对证书业务服务的测试主要是结合业务应用系统的特点，分析业务系统运行实际面临的威胁，验证证书业务服务系统的功能和性能是否满足需求。

对证书业务服务系统的功能测试主要从以下几个方面进行：

- 证书认证系统是否采用国家密码主管部门审批的签名算法完成签名操作，是否提供证书的签发和管理、证书撤销列表的签发和管理、证书/证书撤销列表的发布以及证书审核注册中心的设立、审核及管理等功能；
- 按使用对象分类，系统是否能提供人员证书、设备证书、机构证书三种类型的证书；
- 是否可以提供加密证书和签名证书；
- 数字证书格式是否采用 X.509 V4；
- 系统是否提供证书申请、身份审核、证书下载等服务功能；
- 证书申请、身份审核、证书下载等服务是否都可采用在线或离线两种方式；
- 系统是否提供证书认证策略及操作策略管理、自身证书安全管理等证书管理服务。

对证书业务服务系统还要进行性能测试，以验证是否满足用户的需求。一般性能测试需要考虑以下几个方面：

- 检查证书业务服务系统设计的处理性能是否具备可伸缩配置及扩展的能力；
- 关键部分是否采用双机热备份和磁盘镜像；
- 是否满足系统的不间断运行、在线故障修复和在线系统升级。
- 是否满足需求说明中预测的最大数量用户正常访问的需求，并且，是否有 3～4 倍的冗余，如有必要，需要测试系统的并发压力承受能力。

2．证书查询验证服务系统测试

证书查询验证服务系统为应用系统提供证书认证服务，对证书查询验证服务系统的功能测试，从以下几个方面进行：



- 是否能够提供目录管理与证书查询两种主要服务功能；
- 是否能够根据用户网络设置需求，采用集中式与分布式两种方式构建目录管理服务；
- 系统是否能提供证书和证书撤销列表的发布、证书和证书撤销列表的下载、证书和证书撤销列表的更新、证书和证书撤销列表的恢复、基于 LDAP 技术的目录访问控制服务、目录查询等安全服务功能；
- 是否提供基于 OCSP 技术的证书在线状态查询服务。

同时，如有必要，还需进行下面的性能测试和评估：

- 系统是否满足一定数量的基本的 LDAP 查询和 OCSP 查询并发；
- 是否具备可伸缩配置及扩展能力；
- 是否能为应用系统提供 LDAP、OCSP 查询接口和 C/C++/C#和 Java 等接口。

3．密钥管理系统测试

密钥管理系统是整个安全系统的基础，对密钥管理系统的测试需要考虑以下几点：

- 是否制定了密钥管理策略；
- 是否具备密钥生成、密钥发送、密钥存储、密钥查询、密钥撤消、密钥恢复等基本功能；
- 密钥库管理功能是否完善；
- 密钥管理中心的系统、设备、数据、人员等安全管理是否严密；
- 密钥管理中心的审计、认证、恢复、统计等系统管理是否具备；
- 密钥管理系统与证书认证系统之间是否采用基于身份认证的安全通信协议。

对于大型系统，还需考虑进行以下的性能测试：

- 针对按实际需要配置密钥管理系统，利用企业级的并发压力测试工具测试受理点连接数、签发在用证书数目、密钥发放并发请求数是否满足基本业务需求；
- 测试是否具备系统所需最大量的密钥生成、存储、传送、发布、归档等密钥管理功能；
- 测试是否支持密钥 5 年（或用户要求的年限）保存期；
- 是否具备异地容灾备份；
- 是否具备可伸缩配置及扩展能力；
- 关键部分是否采用双机热备份和磁盘镜像。

4．密码服务系统测试

通常在客户端和服务器端都要提供密码服务。对密码服务系统的测试包括：

- 是否具备基础加解密服务功能；

- 能否为应用提供相对稳定的统一安全服务接口；
- 能否提供对多密码算法的支持；
- 随着业务量的逐渐增加，是否可以灵活地增加密码服务模块，实现性能平滑扩展，且不影响上层的应用系统。

对密码服务系统的性能测试有以下方面。

- RSA 算法密钥长度能否达到 1024~2048 位；ECC 算法密钥长度能否达到 192 位；
- 如有必要进行系统速度测试，对应用层的服务器端密码设备测试项包括：公钥密码算法签名速度、公钥密码算法验证速度、对称密钥密码算法加解密速度；对应用层的客户端密码设备测试项包括：公钥密码算法签名速度、公钥密码算法验证速度、对称密钥密码算法加解密速度，验证是否满足需求；
- 处理性能如公钥密码算法签名等是否具有扩展能力。

5. 可信授权服务系统测试

授权服务系统在信任服务系统的基础上，为应用提供资源的授权管理及访问控制服务，有集中式授权服务与分布式授权服务两种工作模式。

对集中式授权服务的测试，主要验证其是否具有用户管理、审核管理、资源管理、角色管理等主要功能。

对分布式授权服务的测试主要验证其是否有资源访问的签名授权、授权管理等功能。

如有必要，对可信授权服务系统的测试还要考虑，利用并发压力测试工具验证系统是否满足一定的授权服务并发数，以及系统能否提供客户端签名授权和服务器端资源访问授权验证的应用接口，包括 C/C++及 Java 等。

6. 可信时间戳服务系统测试

可信时间戳服务为业务处理的不可抵赖性和可审计性提供支持。对可信时间戳服务的测试点包括：

- 能否从可信时间源（授时中心）获取时间，校准时间戳服务器的时间；
- 能否安全保存时间戳及相关信息；
- 采用公钥技术，能否正确签发可信的时间戳。

一般对可信时间戳服务的测试，还考虑以下性能测试：

- 时间精度能否达到 0.1 秒或用户的需求水平；
- 验证系统时间戳并发请求数能否达到设计要求；
- 相应服务单元是否具有可扩展性。

7. 网络信任域系统测试

通过安全审查的网络可信接入设备和网络信任域管理系统组成网络信任域，测试点



包括：

- 系统是否具备接入认证交换机，确保只有合法的才能接入网络；
- 是否具备网络信任域管理系统，进行网络接入管理配置。

对性能的测试一般包括以下内容。

- 测试认证时间是否小于 1 秒或用户的需求水平；
- 测试接入认证并发数是否满足用户需求；
- 验证接入认证交换机是否提供与客户端通信的遵循 IEEE 802.1X 接口，传输 IEEE 802.1X 认证数据包；
- 验证是否提供网络信任域管理系统与接入认证交换机间的交互接口。

8．故障恢复与容灾备份测试

应用系统的稳定性与可靠性，在一定程度上取决于故障恢复和容灾备份措施。一般有以下三个测试点。

- 故障恢复。整个系统是否存在单点故障；对于关键性应用系统，当任何一台设备失效时，按照预先定义的规则是否能够快速切换；是否采用磁盘镜像技术，实现主机系统到磁盘系统的高速连接；
- 数据备份。对于关键的业务，是否具备必要的热备份机制，例如双机热备、磁盘镜像等；对于所有业务，是否提供磁带备份和恢复机制，保证系统能根据备份策略恢复到指定时间的状态；
- 容灾备份。可否建立异地容灾备份中心，当主中心发生灾难性事件时，由备份中心接管所有的业务；备份中心是否有足够的带宽确保与主中心的数据同步，有足够的处理能力来接管主中心的业务，能否确保快速可靠地与主中心的应用切换。

9．安全产品的选择

安全产品是安全系统构架的基础，合理地选择安全类产品是至关重要的，我们给出以下几点建议。

- 产品是否具有公安部《计算机信息系统安全专用产品销售许可证》和检测报告；
- 如果安全产品涉及数据加密，那么该产品的加密算法是否具有国密办的批准文号；
- 建议尽量选择具有中国自主知识产权的产品，有利于厂家对产品的升级与维护；
- 建议尽量选择具有更先进技术的安全产品；
- 必须考虑安全产品的系统处理速度，因为处理速度直接影响到应用的效果；
- 建议察看安全产品在权威机构（如公安部安全产品检测中心）的检测报告，考察其各项安全功能是否达到安全要求。

11.4　安全性测试方法

11.4.1　功能验证

功能验证是采用软件测试当中的黑盒测试方法，对涉及安全的软件功能，如：用户管理模块、权限管理模块、加密系统、认证系统等进行测试，主要是验证上述功能是否有效，具体方法请参见黑盒测试方法。

11.4.2　漏洞扫描

安全漏洞扫描通常都是借助于特定的漏洞扫描器完成的。漏洞扫描器是一种自动检测远程或本地主机安全性弱点的程序。通过使用漏洞扫描器，系统管理员能够发现所维护信息系统存在的安全漏洞，从而在信息系统网络安全保卫战中做到“有的放矢”，及时修补漏洞。按常规标准，可以将漏洞扫描器分为两种类型：主机漏洞扫描器（Host Scanner）和网络漏洞扫描器（Network Scanner）。主机漏洞扫描器是指在系统本地运行检测系统漏洞的程序，如著名的 COPS、Tripewire、Tiger 等自由软件。网络漏洞扫描器是指基于网络远程检测目标网络和主机系统漏洞的程序，如 Satan、ISS Internet Scanner 等。

安全漏洞扫描是可以用于日常安全防护，同时可以作为对软件产品或信息系统进行测试的手段，可以在安全漏洞造成严重危害前，发现漏洞并加以防范。

11.4.3　模拟攻击试验

对于安全测试来说，模拟攻击试验是一组特殊的黑盒测试案例，我们以模拟攻击来验证软件或信息系统的安全防护能力，下面简要列举在数据处理与数据通信环境中特别关心的几种攻击。在下列各项中，出现了“授权”与“非授权”两个术语。“授权”意指“授予权力”，包含两层意思：这里的权力是指进行某种活动的权力（例如访问数据）；这样的权力被授予某个实体、代理人或进程。于是，授权行为就是履行被授予权力（未被撤销）的那些活动。

- 冒充：就是一个实体假装成一个不同的实体。冒充常与某些别的主动攻击形式一起使用，特别是消息的重演与篡改。例如，截获鉴别序列，并在一个有效的鉴别序列使用过一次后再次使用。特权很少的实体为了得到额外的特权，可能使用冒充成具有这些特权的实体，举例如下。

① 口令猜测：一旦黑客识别了一台主机，而且发现了基于 NetBIOS、Telnet 或 NFS 服务的可利用的用户账号，并成功地猜测出了口令，就能对机器进行控制。



② 缓冲区溢出：由于在很多的服务程序中大意的程序员使用类似于“strcpy()，strcat()”不进行有效位检查的函数，最终可能导致恶意用户编写一小段程序来进一步打开安全缺口，然后将该代码放在缓冲区有效载荷末尾，这样，当发生缓冲区溢出时，返回指针指向恶意代码，执行恶意指令，就可以得到系统的控制权。

- 重演：当一个消息或部分消息为了产生非授权效果而被重复时，出现重演。例如，一个含有鉴别信息的有效消息可能被另一个实体所重演，目的是鉴别它自己（把它当作其他实体）。
- 消息篡改：数据所传送的内容被改变而未被发觉，并导致非授权后果，如下所示。

① DNS 高速缓存污染：由于 DNS 服务器与其他名称服务器交换信息的时候并不进行身份验证，这就使得黑客可以加入不正确的信息，并把用户引向黑客自己的主机。

② 伪造电子邮件：由于 SMTP 并不对邮件发送者的身份进行鉴定，因此黑客可以对内部客户伪造电子邮件，声称是来自某个客户认识并相信的人，并附上可安装的特洛伊木马程序，或者是一个指向恶意网站的链接。

- 服务拒绝：当一个实体不能执行它的正常功能，或它的动作妨碍了别的实体执行它们的正常功能的时候，便发生服务拒绝。这种攻击可能是一般性的，比如一个实体抑制所有的消息，也可能是有具体目标的。例如，一个实体抑制所有流向某一特定目的端的消息，如安全审计服务。这种攻击可以是对通信业务流的抑制，或产生额外的通信业务流。也可能制造出试图破坏网络操作的消息，特别是如果网络具有中继实体，这些中继实体根据从别的中继实体那里接收到的状态报告，来作出路由选择的决定。拒绝服务攻击种类很多，举例如下。

① 死亡之 ping （Ping of Death）：由于在早期的阶段，路由器对包的最大尺寸都有限制，许多操作系统对 TCP/IP 栈的实现在 ICMP 包上都规定为 64KB，并且在读取包的标题头之后，要根据该标题头里包含的信息来为有效载荷生成缓冲区。当产生畸形的、声称自己的尺寸超过 ICMP 上限，也就是加载尺寸超过 64K 上限的包时，就会出现内存分配错误，导致 TCP/IP 堆栈崩溃，致使接受方宕机。

② 泪滴（Teardrop）：泪滴攻击利用那些在 TCP/IP 堆栈实现中信任 IP 碎片中的包的标题头所包含的信息来实现自己的攻击。IP 分段含有指示该分段所包含的是原包的哪一段的信息，某些 TCP/IP（包括 Service Pack 4 以前的 NT）在收到含有重叠偏移的伪造分段时将崩溃。

③ UDP 洪水（UDP Flood）：各种各样的假冒攻击利用简单的 TCP/IP 服务，如 Chargen 和 Echo 来传送毫无用处的数据以占满带宽。通过伪造与某一主机的 Chargen 服务之间的一次的 UDP 连接，回复地址指向开着 Echo 服务的一台主机，这样就生成在两台主机之

间的足够多的无用数据流，如果数据流足够多，就会导致带宽的服务攻击。

④ SYN 洪水（SYN Flood）：一些 TCP/IP 栈的实现，只能等待从有限数量的计算机发来的 ACK 消息，因为它们只有有限的内存缓冲区用于创建连接，如果这一缓冲区充满了虚假连接的初始信息，该服务器就会对接下来的连接请求停止响应，直到缓冲区里的连接企图超时为止。在一些创建连接不受限制的实现里，SYN 洪水也具有类似的影响。

⑤ Land 攻击：在 Land 攻击中，一个特别打造的 SYN 包的原地址和目标地址都被设置成某一个服务器地址，这将导致接受服务器向它自己的地址发送 SYN-ACK 消息，结果，这个地址又发回 ACK 消息并创建一个空连接，每一个这样的连接都将保留，直到超时。各种系统对 Land 攻击的反应不同，许多 UNIX 实现将崩溃，NT 变得极其缓慢（大约持续 5 分钟）。

⑥ Smurf 攻击：一个简单的 Smurf 攻击，通过使用将回复地址设置成受害网络的广播地址的 ICMP 应答请求（ping）数据包，来淹没受害主机的方式进行，最终导致该网络的所有主机都对此 ICMP 应答请求作出答复，导致网络阻塞，比“Ping of Death”洪水的流量高出一或两个数量级。更加复杂的 Smurf 将源地址改为第三方的受害者，最终导致第三方雪崩。

⑦ Fraggle 攻击：Fraggle 攻击对 Smurf 攻击作了简单的修改，使用的是 UDP 应答消息，而非 ICMP。

⑧ 电子邮件炸弹：电子邮件炸弹是最古老的匿名攻击之一，通过设置一台机器，不断大量地向同一地址发送电子邮件，攻击者能够耗尽接收者网络的带宽。

⑨ 畸形消息攻击：各类操作系统上的许多服务都存在此类问题，由于这些服务在处理信息之前没有进行适当正确的错误校验，在收到畸形的信息时可能会崩溃。

- 内部攻击：当系统的合法用户以非故意或非授权方式进行动作时就成为内部攻击。多数已知的计算机犯罪都和使系统安全遭受损害的内部攻击有密切的关系。能用来防止内部攻击的保护方法包括：所有管理数据流进行加密；利用包括使用强口令在内的多级控制机制和集中管理机制来加强系统的控制能力；为分布在不同场所的业务部门划分 VLAN，将数据流隔离在特定部门；利用防火墙为进出网络的用户提供认证功能，提供访问控制保护；使用安全日志记录网络管理数据流等。
- 外部攻击：外部攻击可以使用的办法有：搭线（主动的与被动的）、截取辐射、冒充为系统的授权用户，冒充为系统的组成部分、为鉴别或访问控制机制设置旁路等。
- 陷阱门：当系统的实体受到改变，致使一个攻击者能对命令或对预定的事件或



事件序列产生非授权的影响时，其结果就称为陷阱门。例如，口令的有效性可能被修改，使得除了其正常效力之外也使攻击者的口令生效。

- 特洛伊木马：对系统而言的特洛伊木马，是指它不但具有自己的授权功能，而且还有非授权功能。一个向非授权信道拷贝消息的中继就是一个特洛伊木马。典型的特洛伊木马有 NetBus、BackOrifice 和 BO2k 等。

11.4.4 侦听技术

侦听技术实际上是在数据通信或数据交互过程，对数据进行截取分析的过程。目前最为流行的是网络数据包的捕获技术，通常我们称为 Capture，黑客可以利用该项技术实现数据的盗用，而测试人员同样可利用该项技术实现安全测试。

该项技术主要用于对网络加密的验证。

11.5 软件产品安全测试

软件产品安全测试侧重于以下方面：用户对数据或业务功能的访问控制，数据存储和数据通信的远程安全控制。

11.5.1 用户管理和访问控制

1. 用户权限控制

对于应用软件来说，用户权限的控制是比较重要的，一个用户能够访问一个应用系统的权限，通常我们认为来源于三个方面，即：应用软件本身、操作系统和数据库。应用软件产品在开发的过程中，主要采用用户名和密码登录的方式完成。对于安全强度较高的软件也可采用指纹认证、智能卡认证等方式进行。对于测试者来说，应当注意以下几个方面。首先应当评价用户权限控制的体系合理性，是否采用三层架构的管理模式，即系统管理员、业务领导和操作人员三级分离；用户名称基本采用中文和英文两种，对于测试来说，对于用户名称的测试关键在于测试用户名称的惟一性。惟一性的体现基本基于以下两个方面。

① 同时存在的用户名称在不考虑大小写的状态下，不能够同名；

② 对于关键领域的软件产品和安全性要求较高的软件，应当同时保证使用过的用户在用户删除或停用后，保留该用户记录，并且新用户不得与该用户同名。

用户口令应当满足当前流行的控制模式，注意测试用户口令的强度和口令存储的位置和加密强度，如下所示。

① 最大口令时效：指定用户可以保留当前口令的时间。

② 最小口令时效：指定在修改口令之前，用户必须保留口令的时间。

③ 口令历史：确定系统将要记住的口令的数量。如果用户选择的口令存在于口令历史数据库中，系统将强制用户选择其他口令。

④ 最小口令长度：对于用户口令，可以包含的最少的可以接受的字符数目。

⑤ 口令复杂度：在口令中要求用户使用非字母数字的字符或大写字母。

⑥ 加密选项：可以加密本地存储的口令。

⑦ 口令锁定：口令锁定是被用来对付猜测口令的主要工具。在输入的非法口令达到规定的次数之后，系统将禁用这个账户。这种技术在对付远程暴力攻击的时候特别有效。

⑧ 账户复位：账户锁定后定义是否可以在规定时间间隔后自动恢复，可以减轻系统管理员的工作强度。

用户权限分配方式是否合理，用户认证的目的在于访问控制，对于软件产品来说，主要是用户能够使用某些功能或访问某些数据的能力。从软件测试的角度来说，应当结合黑盒功能测试，首先测试软件用户权限系统本身权限分配的细致程度，比如，对于查询功能来说，查询的数据是否能够按照业务需求进行细致划分，然后对特定权限用户访问系统功能的能力进行测试，该部分测试应当充分利用等价值划分方法进行案例设计，避免重复测试。

2．操作系统安全性的测试

操作系统本身的安全性对应用软件存在影响，测试中要考察：是否关闭或卸载了不必要的服务和程序；是否存在不必要的账户；权限设置是否合理；安装相应的安全补丁程序的情况；操作系统日志管理等。

3．数据库权限的测试

数据库权限管理目前存在较多问题，该部分的测试有如下三个方面。

① 应用软件部署后，数据库管理用户的设置应当注意对账号的保护，超级用户的口令不得为空或默认口令。对数据库的账号和组的权限作相应设置，如锁定一些默认的数据库用户；撤销不必要的权限。

② 数据库中关于应用软件用户权限和口令存储的相关表格，尽量采用加密算法进行加密。

③ 软件企业在进行软件产品开发时，开发人员通常为了开发方便，在客户端与数据库通信时，均使用超级用户及默认密码（例如：username="sa"，password=""）访问数据库，这种方式将会带来严重的安全隐患，测试人员可以通过网络侦听的技术，或使用白盒测试方法进行测试，并且应当建议开发者，根据不同程序访问数据库的功能，使用不同的数据库用户进行连接，且必须设置复杂的密码。



11.5.2 通信加密

通信加密是保证数据在传输过程中数据的保密性和一致性的测试，软件产品在技术上通常使用链路加密、数据加密的方式进行，目前使用的加密技术包括 VPN 技术、对称加密算法、非对称加密算法、HASH 算法等。

VPN 技术通常使用 Kebores 加密算法结合 IPSec 加密协议，实现通信链路的加密，加密强度较高，多用于广域网中的数据安全传输，但是操作较复杂。

对称加密算法和非对称加密算法主要包括 RSA、DSA（非对称）、DES（对称），使用上述算法的主要目的在于解决数据的保密性传输。

HASH 算法的目的在于保护数据内容的一致性，防止数据在传输过程中被篡改。

通信加密测试的目的主要是，保证软件产品确实在开发过程中，按照设计要求使用了上述方法进行了通信的加密，通常使用验证和侦听技术完成。另外还需要注意的是，如果开发过程中使用上述标准对加密函数进行加密，加密强度是受密钥的复杂度等因素影响的，无须对本身算法加密强度进行论证，如果使用自定义的加密函数，测试人员应当使用破解技术对算法本身的加密强度进行论证，或将自定义加密算法提交特定机构进行测试和认证。

11.5.3 安全日志测试

我们上面已经提到，安全日志是软件产品被动防范的措施，是重要的安全功能，因此需要在软件测试过程中重点测试。

日志应当记录所有用户访问系统的操作内容，包括登录用户名称、登录时间、浏览数据动作、修改数据动作、删除数据动作、退出时间、登录机器的 IP 等。

测试人员应该根据业主要求或设计需求，对日志的完整性、正确性进行测试，测试安全日志是否包含上述全部内容，是否正确，并且对于大型应用软件来说，系统是否提供了安全日志的智能统计分析能力，是否可以按照各种特征项进行日志统计，分析潜在的安全隐患，并且及时发现非法行为。

第 12 章　兼容性测试

12.1　兼容性测试概述

兼容性测试将验证软件与其所依赖的环境的依赖程度，包括对硬件的依赖程度，对平台软件、其他软件的依赖程度等，本章对兼容性测试的范围和方法进行了描述，在此基础上，对数据的兼容性、特别是新旧系统割接时数据的迁移测试进行了深入的描述。

12.2　兼容性测试环境的准备

兼容性测试需要在各种各样的软硬件环境下进行，因此兼容性测试是软件测试中投入较大的一部分。测试中的硬件环境指进行测试所必需的服务器、客户端、网络连接设备，以及打印机、扫描仪等辅助硬件设备所构成的环境；软件环境则指被测软件运行所需的操作系统、数据库、中间件、浏览器及与被测软件共存的其他应用软件等构成的环境。

如果在计划兼容性测试时遇到测试环境准备上的问题，可以尝试通过以下几种渠道解决。

- 向硬件厂商租用或借用。硬件厂商如果有展示厅或测试中心，一般会欢迎其他厂商使用他们的设备，因为这样同样测试了其硬件对软件的兼容性。
- 采用试用版软件。软件可采用试用版，但需注意其限制条件，如连接数、有效时间等，有时这些限制条件会导致软件测试不充分。
- 在条件完善的专业测试实验室里完成测试。目前已经出现越来越多的专业评测机构，如中国软件评测中心等。这些机构往往拥有大量的软硬件设备和测试工具，可以协助软件开发商完成各类测试。

12.3　硬件兼容性的测试

12.3.1　硬件兼容性测试的目的

所有软件都需向用户说明其运行的硬件环境，对于多层结构的软件系统来说，需要



分别说明其服务器端、客户端以及网络所需的环境。兼容性测试的目的就是确认这些对于硬件环境的描述是否正确、合理。

硬件兼容性测试需确认以下几点。

- 最低配置是否能够满足系统运行的需要。在最低配置下，所有的软件功能必须能够完整地实现，软件运行速度、响应时间应在用户能够忍受的范围内。
- 在推荐配置下系统的响应迅速。应当注意的是，推荐配置必须合理，一味地追求高配置，一方面可能掩盖软件的性能缺陷，另一方面限制了软件的应用范围，也是不合理的。
- 考察软件对运行硬件环境有无特殊说明，如对 CPU、网卡、声卡、显卡型号等有无特别声明。有些软件可能在不同的硬件环境中出现不同的运行结果或是在某些环境下根本就不能执行，如操作系统或数据库软件能否支持多个 CPU 协同工作，对内存的多少是否过于敏感等。
- 为了满足不同的使用需求，软件系统能否运行在多种硬件配置环境下，并且系统功能和性能都能满足设计需求。这样的测试为企业的硬件选型与部署提供了依据。例如，是否可将 Web 服务器与数据库服务器部署在一台物理服务器上，或者需要将二者分别部署在不同的物理服务器上；再如，软件系统的客户端与服务器是否可采用企业专用网络通信，或者提供 Modem 拨号上网通信。

12.3.2 与整机的兼容性

整机兼容性测试将确认软件要求的最低配置和推荐配置的合理性和正确性。主要指标包括对机型的要求和对 CPU、内存、硬盘的要求，其他还包括对 RAID 的支持、对 SCSI 的支持等。

- CPU。各类软件的最低配置和推荐配置都必须注明对 CPU 的要求，包括 CPU 类型和主频的要求。大多数软件仍以对 Intel X86 体系的支持为主，Intel 主流 CPU 的性能差别主要体现在频率、缓存两方面。其中频率因素又分为主频与外频两部分，缓存因素分为缓存容量、缓存速度、缓存潜伏时间三部分，例如，由于二级缓存的不同，选用 Celeron 处理器还是 PIII 处理器会对速度测试结果造成一定的影响，因此配置测试环境时需特别注意。应用软件及客户端软件对 CPU 的推荐配置要求应当比目前主流 CPU 略低，服务器端的最低配置必须能够使软件正常工作，推荐配置应保证软硬件构成的系统在正常业务的压力负载下，CPU 资源占用平均值不超过 75%。

  内存。所有软件都希望运行环境中的内存越大越好，内存容量直接影响着软件的运行效率。对于一般软件的测试，只需设计不同内存容量下的测试，但在有

些情况下，也需考虑内存参数的设置对性能的影响，这主要集中在 CAS（Column Address Strobe，列地址选通脉冲）、RAS（Row Address Strobe，行地址选通脉冲）、RAS-CAS 等参数的设置上。

- 硬盘。一般只有对特殊软件的硬件兼容性测试需考虑硬盘的转速、缓存容量、寻址时间等参数。但如果将软件速度与硬件兼容性测试结合进行，则需要记录这些硬盘参数。

12.3.3 与板卡及配件的兼容性

在各类软件中，与板卡及配件打交道的是操作系统和驱动程序。其中，桌面操作系统的硬件兼容性无疑是最复杂的，它涉及几乎所有类型的板卡和配件。随着自由软件的蓬勃发展，桌面操作系统成为了新的软件开发热点。

其他软件也会对某些配件比较敏感，如游戏软件对显卡，播放器对显卡和声卡，传真软件对调制解调器等。

一般情况下，板卡和配件与软件的兼容性需要考虑以下几个方面。

- 独立板卡。各类板卡技术的迅速发展使操作系统的支持变得越来越困难。由于微软的 Windows 操作系统在桌面领域占有率高，因此板卡生产商一般会提供 Windows 上的驱动程序，而对其他操作系统则较少顾及，因此其他操作系统往往需做大量工作才能达到与这些设备兼容。而操作系统与各类板卡的兼容不仅仅体现在“能用”上，还必须达到“好用”。
- 主板芯片组。BX、810 系列、815 系列、GX 芯片组设计上的差异要求测试中必须分别进行测试。同时，由于针对不同芯片组开发的驱动程序效率不同，也可能造成与预期效果不同的测试结果。对于集成主板，由于各部分的相互影响，往往会对被测桌面操作系统的兼容性造成挑战。即使主板上集成的声卡、显卡等的芯片组都能够分别被支持，整个主板也需经过测试才能列入兼容性列表。
- 驱动程序中的自由软件。即使从互联网上可以找到很多硬件驱动的自由软件，但这些软件大多是由爱好者开发的，由于爱好者往往不能够完整地了解硬件产品的核心技术，而且没有严密组织的开发过程，造成驱动程序中存在众多缺陷。如果被测软件中存在这样的情况就要特别注意。

12.3.4 与打印机的兼容性

打印机兼容性的测试对于操作系统、办公软件、网站和其他打印功能比较重要的软件来说，都是不可缺少的。

对于不同厂商、不同型号的打印机需要分别进行测试。



打印机兼容性测试主要包括以下项目。

- 安装或能够调用系统安装的打印机；
- 能打印测试页；
- 能选择不同幅面的纸张；
- 能选择打印精度（打印分辨率）；
- 纸张横、纵打印调整功能；
- 逐页打印功能；
- 多份打印功能；
- 可以进行打印机的维护（比如更换墨水、清洗打印头等）；
- 具备双面打印器的打印机能够实现自动双面打印功能；
- 网络打印机能够实现网络打印功能。

12.3.5 其他

软件运行所依赖的其他所有硬件设备，都应进行兼容性测试，如操作系统与红外鼠标、键盘的兼容性，识别软件对扫描仪的兼容性，视频编辑软件与 Jog 搜索轮的兼容性，刻录软件对光驱的兼容性等，需根据实际情况确定测试范围和测试方法。

12.4 软件兼容性测试

12.4.1 与操作系统的兼容性

由于软件开发技术的限制以及各种操作系统之间存在着巨大的差异性，因此，目前大多商业软件并不能达到理想的平台无关性。如果该软件承诺可以在多种操作系统上运行，那么我们就需要测试它与操作系统的兼容性。对于多层体系结构的软件，要分别考虑前端和后端操作系统的可选择性。

操作系统兼容性的测试内容不仅包括安装，还需对关键流程进行检查。需要测试哪些操作系统上的兼容性，首先取决于软件用户文档上对用户的承诺，其次就要考虑以下几个问题。

- Windows 平台：随着微软对 Windows 平台的不断升级，对于上一代操作系统，如 Windows 32、Windows 95、Windows NT4，除非有特殊需求，一般都不再作出支持承诺，一些软件甚至不对 Windows 98 进行承诺。对于 B/S 结构的客户端，至少需在 Windows 98、Windows ME、Windows 2000、Windows XP 上进行测试，英文版和中文版需分别测试，在英文版操作系统上测试中文版软件时，要特别

注意是否会出现英文信息或乱字符，在中文版操作系统上测试英文软件时，注意是否存在提示文字不能完全显示的现象。测试前要保证测试环境中所有的补丁都已安装，在用户文档中也应给出提示。如果有必要进行更严格的测试，则可以增加对不同版本补丁的兼容性测试。

- Linux 平台：Linux 作为自由软件，其核心版本是惟一的，而发行版本则不受限制。从 RedHat、Turbo Linux 到国内的中科红旗、中软等，版本之间存在着较大的差异。因此被测软件不能简单地说是支持 Linux，测试也不能只在 RedHat 最新发行版本上进行，需要对多发行商、多版本进行测试，用户文档中的内容应明确至发行商和版本号，不能笼统地描述为支持 Linux 平台。
- UNIX 平台：与 Linux 平台一样，UNIX 平台也存在着 Solaris、IBM、HP 等多厂商的多版本，不过由于在这些 UNIX 平台上运行的软件往往至少需要重新编译才能运行，所以只需按软件的承诺选择测试环境即可。
- Macintosh：使用这类系统的往往是图形专用软件。对于 Web 站点也需要进行 Macintosh 系统下的测试，有些字体在这个系统下可能不存在，因此需要确认选择了备用字体。

12.4.2 与数据库的兼容性

20 世纪 90 年代以后，数据库的应用向多元化方向发展，大型的应用往往涉及不同的应用领域，需要不同模型的数据库，同时，各个数据库管理系统之间的互操作性、移植性都越来越受到大家的重视，再加上开发工具的发展，促进了数据库标准的成熟与发展。数据库标准主要包括：SQL、ODBC、JDBC、ADO、OLE DB、JDO（Java Data Object）等。

SQL（Structured Query Language）是对数据库进行操作的基本语言，SQL 于 1974 年提出，1986 年 10 月成为数据语言的美国标准，1987 年成为 ISO 标准，以后进行多次版本升级，到目前为止，已经制定出的 SQL 标准有 SQL-86、SQL-89、SQL-92 和 SQL-99，目前市场普遍接受的是 SQL-92 标准。SQL-92 分为 4 个一致性等级（其中 3 个级是 ANSI 标准中定义的，还有 1 个级是由 NIST 在作符合性测试时定义的），分别是入门级、过渡级、中间级和完全级。目前数据库产品对 SQL 标准的支持程度并不相同。

ODBC（Open Data Base Connectivity，开放数据库互连）是微软公司开发的一套开放数据库系统应用程序接口规范，它是微软公司 WOSA（Windows Open System Architecture，即 Windows 开放系统体系结构）的主要组成部分。ODBC 接口的最大优点是其互操作能力强，理想情况下，每一个驱动程序和数据源应支持完全相同的 ODBC 函数调用和 SQL 语句，使得 ODBC 应用程序可以操作所有的数据库系统。然而，不同的数据库系统对 SQL 语法的支持程度各不相同，实现的 ODBC 规范所定义的功能也会有

所不同。

JDBC (Java Data Base Connectivity，Java 数据库连接)，目前，JDBC 已经推出了 1.0、2.0、3.0 三个版本，同样，各个数据库对 JDBC 的支持也并不相同。

总之，数据库虽然有各种标准，但是由于各个数据库对标准的支持程度并不相同，基于一种数据库开发的应用系统，在另外一种数据库上未必运行良好，而现在很多软件需考虑对不同数据库平台或同一数据库的不同版本的支持能力，如从 Sybase 平台迁移到 Oracle 平台，从 Oracle 8i 升级到 Oracle 9i 等，这就要求我们必须做数据库兼容性测试工作。

此类测试可能是主动的，在软件开发阶段就已进行，也可能是被动的，由于新版本的出现或用户的需求改变而被迫进行。

数据库兼容性测试要点如下。

- 完整性测试。检查原数据库中各种对象是否全部移入新数据库，同时比较数据表中数据内容数是否相同。
- 应用系统测试。模拟普通用户操作应用的过程，对应用进行操作并检查运行结果，从以往的测试经验来看，如果开发中使用了存储过程，那么在数据库移植时最容易出现问题。
- 性能测试。上两项测试通过后，针对服务器、数据库进行性能测试，并与在原数据库下记录的性能基准数据进行比照，找出性能方面的问题，并有针对性地进行性能优化。

12.4.3 与中间件的兼容性

涉及中间件的系统一般已不是一个单纯的软件，而是一个有一定规模的系统了。中间件作为其中最关键的部件之一，许多开发都与其紧密相连，所以，中间件的更换并不常见，中间件兼容性的测试通常情况下是指对不同版本、不同补丁包的兼容性。如：系统中的中间件 WebSphere 从 Sp2 升级到 Sp3 时，需检查应用是否能够正确运行，性能是否有所提高或至少保持不变。

中间件兼容性的测试方法与数据库兼容性的测试方法大同小异，这里不再赘述。

12.4.4 与浏览器的兼容性

浏览器是 Web 客户端最核心的构件，被测软件的客户端能否使用 Netscape、Internet Explorer 或 Lynx 进行浏览呢？有些 HTML 标签或脚本只能在某些特定的浏览器上显示。应当确认图片有替代文字，因为可能会有用户使用文本浏览器。如果使用了 SSL 安全特性，则需要关注浏览器的版本，因为老版本可能不支持 SSL，应对老版本的用户给出相

关的提示信息。

来自不同厂商的浏览器对 Java、JavaScript、ActiveX、plug-ins 或不同版本的 HTML 有不同的支持。例如，ActiveX 是 Microsoft 的产品，是为 Internet Explorer 而设计的，JavaScript 是 Netscape 的产品，Java 是 Sun 的产品等。另外，框架和层次结构风格在不同的浏览器中也有不同的显示，甚至根本无法显示。不同的浏览器对安全性和 Java 的设置也不一样。

测试浏览器兼容性的一个方法是创建一个兼容性矩阵。在这个矩阵中，测试不同厂商、不同版本的浏览器对某些构件和设置的适应性。

表 12-1 是一个对浏览器兼容性进行测试的记录表样本。

表 12-1　浏览器兼容性测试记录表

	Applets	JavaScript	ActiveX	VBScript
Internet Explorer 5.X				
Internet Explorer 6.X				
Netscape Navigator 5.X				
Netscape Navigator 6.X				

12.4.5　与其他软件的兼容性

除了以上各项软件的兼容性以外，我们还需要考虑以下问题。

- 与支持软件的兼容性。软件运行还需要哪些应用软件支持？ERP 软件可以仅提供财务软件接口，而本身并不包含财务软件；财务软件也可以不包含表格处理模块，而调用其他表处理软件。这些被测软件运行所必须的其他软件都应当进行兼容性测试，测试中要对其所依赖的软件的各个版本分别进行测试。
- 与其他同类软件的兼容性。对于通用软件来说，这是一个重要问题。由于通用软件应用范围广，开发商多，功能重复性高，在系统中可能会要求相同的系统资源，造成注册表冲突等问题，因此需要进行兼容性测试，以判断与其他同类软件安装在同一系统上、同时使用，是否会造成其他软件运行错误，或本身能否正确实现其功能，例如，测试杀毒软件时，检查将其与其他多个杀毒软件共同安装于同一系统中的情况。
- 与其他非同类软件的兼容性。例如：测试办公软件时，将其与杀毒软件共同安装于同一系统时，是否会造成软件安装错误或运行错误。如果在杀毒软件运行的状态下不能顺利安装，则需在用户手册中的软件安装部分给出明确提示。



12.5 数据兼容性测试

12.5.1 不同数据格式的兼容性

数据兼容是指软件之间能否正确地交互和共享信息。制定数据兼容性测试用例时可以参考以下几项内容。

- 在被测软件与其他程序间复制粘贴的文字是否正确？带格式的文字呢？表格呢？图形呢？
- 在以前的版本下保存的文字在新的版本中是否能被打开；所有的特点是否都能被保留；包含新特性的新版本文件在旧系统中是否能被打开；新特性在旧版本中将如何解释。
- 被测软件是一个系列软件中的一个吗，与本系列中的软件以何种形式交换数据。
- 与同类软件间能否进行数据交换，软件是否提供对其他常用数据格式的支持。例如，办公软件是否支持常用的微软 Office 或 WPS 等文件格式，支持的程度如何，即软件是否能完全正确地读出这些格式的文件？保存为这些格式的文件呢？
- 测试中需要明确业界有没有针对被测软件内容进行数据交换定义的标准或规范。例如，有些行业要求本行业的专业软件必须能够导入/导出 XML 格式的文件，且必须符合一定的数据格式规范。

12.5.2 XML 符合性

目前的数据格式多种多样，使不同类型的数据交换和集成成为很困难的事情，如：搜索多样的不兼容的数据库实际上是不可能的，不同的办公软件使用不同的格式，因此不能相互打开彼此的文档等。

XML 作为一种较新的技术，能够使不同来源的结构化的数据较容易地结合在一起，提供了一个描述数据和交换数据的有效手段。

XML（Extensible Markup Language,可扩展标记语言）是一种元标记语言，它使用简单灵活的标准格式。XML 主要有 3 个组成元素：Schema（模式）、XSL（可扩展样式语言）和 XLL（可扩展链接语言），其中 Schema 规定了 XML 文件的逻辑结构，定义了 XML 文件中的元素、元素的属性以及元素和元素属性之间的联系，它可以帮助 XML 的分析程序校验 XML 文件标记的合法性；XSL 是用于规定 XML 文档样式的语言，它能

在客户端使 Web 浏览器改变文档的表示法，从而不需要再与服务器进行交互通信；XLL 将进一步扩展目前 Web 上已有的简单链接。

目前，我国中文办公软件标准基本形成了《中文办公软件文档格式规范 XML Schema 内容说明规范》征求意见稿，一些行业软件已将 XML 作为其行业规范进行推荐，并得到了开发商的广泛认可。

XML 测试的需求往往来自于业界已有的数据格式规范，一般是一套 Schema 文件。其测试步骤一般为：

- 在测试工具中建立标准模板；
- 用被测软件按要求导出数据；
- 将导出的数据与标准模板进行对比匹配测试；
- 输出测试结果。

测试中用到的数据比较工具可以采用已有的 XML 解析器如 XMLSPY，或有针对性地开发出一些专用工具。

12.6　平台化软件兼容性测试

12.6.1　平台化软件概述

随着软件系统的规模变得越来越大，平台化软件的使用越来越普遍。平台化软件是指用来构建与支撑应用软件的独立软件系统。软件平台有两个基本要素，即支撑环境和开发体系，其中支撑环境是指应用软件系统开发与运行的基本条件，开发体系是指开发与维护管理应用软件的工具与方法。它又可以分为技术支撑型平台软件和应用实现型平台软件两种类型的平台。平台软件的结构图如图 12-1 所示。

- 技术支撑型平台：为软件系统研发提供通用技术基础架构，主要面向软件开发人员，主要包含应用软件的运行支持体系和上层开发的工具，上层应用系统由用户进行研发。我们熟悉的有 BEA WebLogic、IBM WebSphere，Web 服务器等。
- 应用实现型平台：指用来构建与支撑应用软件的独立软件系统。主要面向应用软件的终端用户。它既要包括应用软件的运行支持体系和上层应用开发工具，又要直接包括上层应用系统。如 ERP 系统平台。

平台化软件具有很多传统软件所不具有的优势，包括以下几项内容。

- 应用的广泛性：平台化软件的功能更加全面、覆盖面更加广泛。如平台化的 ERP 能有效地将财务管理、销售管理、财务管理、客户关系、采购与库存、人力资



源等各种管理模块集成在一起，形成一个庞大的系统。平台化软件的各种信息统一存储，实现管理信息的共享，从而提高各级组织内部有效的协作和快速反应。

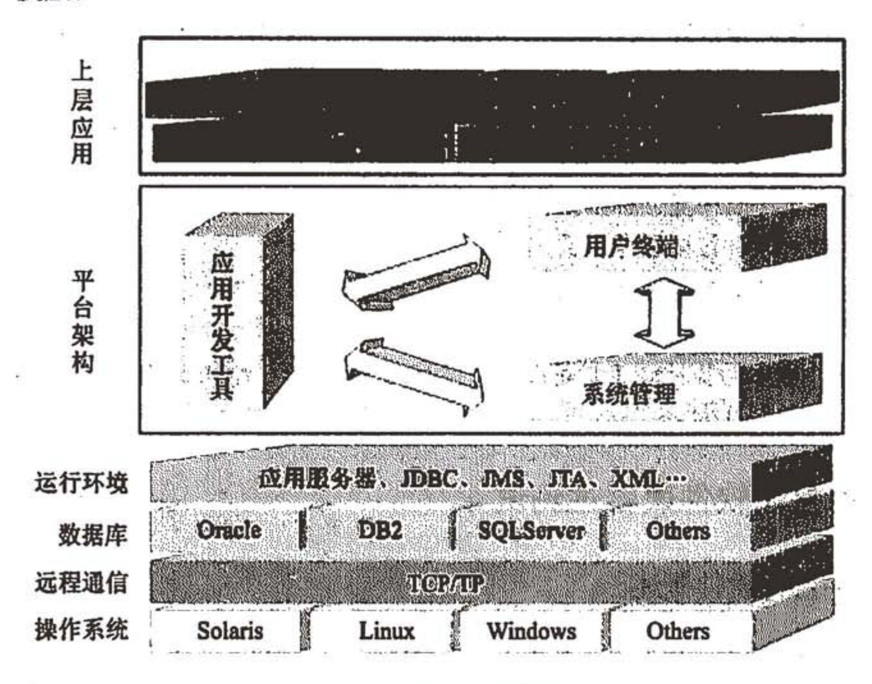


图 12-1 平台软件结构图

- 良好的开发性：平台化软件是建立在共同平台上的一个系统，模块相互之间既具有较强的独立性，又可以独立使用，通过统一的数据接口可实现相互间的无缝集成，同时可以实现一致的对外接口。因此，平台化软件具有更大程度的灵活性和扩展性，不仅可以根据客户当前的需求进行选择和搭配使用，而且具备了更好的二次开发接口。
- 快捷的适应性：平台化软件的上层应用开发工具可以实现免编程的应用系统修改，具有更快捷、方便的适应能力。

平台化软件有着诸多优点，目前平台化软件已被认为是管理软件的发展趋势。设计良好的平台化软件应该可以普遍应用于企业管理系统、校园管理系统、电子政务、医院管理系统等各行各业。真正的平台化产品不应该是在原有的固化的软件基础上的改造，因为原有的系统使用硬写代码的方式实现，无法与新型的平台化软件的运行支撑系统和应用开发工具结合，实现客户个性化需求的免编程定制。新型的平台化产品必须具备两

个基本要素，实现应用的完全可定制，而不是原有系统外围的所谓“二次开发”。

12.6.2 平台化软件的兼容性测试策略

平台化软件在设计上要求具备灵活方便的二次开发能力，实现分布式应用系统，做到硬件环境独立和软件环境独立，实现上层应用的技术无关性以及采用 B/S 与 C/S 结构相结合，因此对于平台软件来说，兼容性测试是极为重要和复杂的。

- 跨硬件平台能力：在不同规模的硬件平台上进行测试，例如，从 32 位机到 64 位机，从单机运行到集群运行。
- 跨操作系统能力：平台化软件应可以部署在各种流行的操作系统上，不仅应当在 Windows 操作系统上进行测试，还应当在 UNIX、Linux、UNIX、MacOS 等系统上进行测试。
- 支持多种数据库系统：包括 SQL Server、Oracle、DB2、Sybase 等企业级数据库，以及达梦、OpenBase、KingBase、Oscar 等国产主流数据库。
- 客户端兼容性测试：平台化软件往往实现的是分布式的应用系统，因此可能采用 C/S 或 B/S 结构。对于需要进行软件安装的客户端，需要对客户端的硬件、软件兼容性分别进行测试，如果采用浏览器，则还需进行对浏览器兼容性的测试。
- 数据兼容性测试：平台化软件是建立在共同平台上的一个系统，模块相互之间既具有较强的独立性，又可以独立使用，通过统一的数据接口又以可实现相互间的无缝集成，同时可以实现一致的对外接口。因此，各模块间数据的兼容性也是平台化软件的测试点之一。

由此可见，平台化软件的测试几乎覆盖了软硬件测试的全部领域，为了将企业原有数据向平台系统转移，还需要用到下面讲到的新旧系统数据迁移测试。

12.7 新旧系统数据迁移测试

12.7.1 新旧系统数据迁移技术

随着技术的发展，原有的系统不断被功能更强大的新系统所取代。在新旧系统的切换过程中，必然要面临一个数据迁移的问题。

有的旧系统从启用到被新系统取代，在其使用期间往往积累了大量珍贵的历史数据，其中许多历史数据都是新系统顺利启用所必须的。另外，这些历史数据也是进行决策分析的重要依据。数据迁移，就是将这些历史数据进行清理、转换，并装载到新系统中的过程。数据迁移主要适用于一套旧系统切换到另一套新系统，或多套旧系统切换到



同一套新系统时，需要将旧系统中的历史数据转换到新系统中的情况。银行、电信、税务、工商、保险以及销售等领域发生系统切换时，一般都需要进行数据迁移。对于多对一的情况，例如，由于信息化建设的先后，造成有多个不同的系统同时运行，但相互间不能做到有效数据共享，需要一套新系统能够解决几套旧系统间的数据共享问题。

数据迁移对系统切换乃至新系统的运行有着十分重要的意义。数据迁移的质量不光是新系统成功上线的重要前提，同时也是新系统今后稳定运行的有力保障。如果数据迁移失败，新系统将不能正常启用；如果数据迁移的质量较差，没能屏蔽全部的垃圾数据，对新系统将会造成很大的隐患，新系统一旦访问这些垃圾数据，可能会由这些垃圾数据产生新的错误数据，严重时还会导致系统异常。

相反，成功的数据迁移可以有效地保障新系统的顺利运行，能够继承珍贵的历史数据。因为无论对于一个公司还是一个部门，历史数据无疑都是一种十分珍贵的资源，例如，公司的客户信息、银行的存款记录、税务部门的纳税资料等。因此，对数据迁移进行充分的测试非常必要。

12.7.2 新旧系统数据迁移的实现与测试

由于系统部署千差万别，数据迁移具有一定的复杂性和风险性，有必要在进行新旧系统数据迁移之前对迁移进行测试，提高迁移成功的概率，降低风险。

数据迁移的实现可以分为三个阶段：数据迁移前的准备、数据迁移的实施和数据迁移后的校验。

由于数据迁移的特点，大量的工作都需要在准备阶段完成，充分而周到的准备工作是完成数据迁移的基础。具体而言，要进行待迁移数据源的详细说明，包括数据的存放方式、数据量、数据的时间跨度等，建立新旧系统数据库的数据字典，对旧系统的历史数据进行质量分析，新旧系统数据结构的差异分析；新旧系统代码数据的差异分析；建立新旧系统数据库表的映射关系，对无法映射字段的处理方法，开发、部属数据转换与迁移工具，编写数据转换的校验程序，制定数据转换的应急措施。

数据迁移的实施是实现数据迁移的三个阶段中最重要的环节。它要求制定数据转换的详细实施步骤流程；准备数据迁移环境；结束未处理完的业务事项，或将其告一段落；对数据迁移涉及的技术进行测试；最后实施数据迁移。

数据迁移后的校验是对迁移工作的检查，数据校验的结果是判断新系统能否正式启用的重要依据。可以通过质量检查工具或编写检查程序进行数据校验，通过试运行新系统的功能模块，特别是查询、报表功能，检查数据的准确性。

为了保证数据的安全性，在测试和实施时还可以考虑以下措施。

- 在实际运行环境之外搭建模拟环境，导入部分或全部数据，在模拟环境中进行

一次或数次模拟迁移尝试。测试不仅要包括旧系统向新系统的迁移，还需进行新系统到旧系统的反向迁移，以确保在迁移过程失败时，可以及时恢复旧系统。在测试过程中还要详细记录遇到的问题，研究解决方法，并预测迁移风险。

- 将现有数据进行备份，检查备份数据的正确性；作两套备份，比较两份备份，以保证备份数据完整可靠。
- 如果有备份系统，则先将备份系统升级到新系统，保持主服务器的旧系统不动，切换至备份服务器运行一周，若一切正常再升级主服务器，升级成功后切换至主服务器运行。

12.8 小结

并不是每个软件都要测试所有的兼容性项目的测试，这里列出的也并不是一本字典，不可能收录所有的测试项，我们既希望列出尽可能多的测试项供参考，又不可能穷尽所有的测试项。在实际测试中，要按照软件类型、需求定位和测试环境进行选择，并以此为思路扩充测试方案。

兼容性测试应当充分验证软件定义的适用范围，为开发者和用户提供软件使用的信心。但由于兼容性测试所需的投入较大，因此测试管理人员必须作出取舍，以较小的投入达到最好的测试效果。

还要注意的是，对于定制系统来说，兼容性测试应尽早进行，否则系统投入使用后，随着系统中数据的增多，兼容性测试的风险和投入将越来越大。



第 13 章　标准符合性测试

13.1　概述

随着信息技术的不断深入和发展，各种系统软件、应用软件越来越多，而它们之间可以相互方便地进行通信和数据交换的需求越来越紧迫。

在我国许多行业领域，在逐步完成了数据信息电子化积累之后，如何有效地进行数据的交换和共享，已经越来越被相关的行业主管部门所重视。没有信息的标准化，各企业、事业单位的数据就是一个孤岛，就不利于整个行业的数据共享和业务整合。

以检察行业为例，在由几千个检察院局域网络系统组成的检察信息网络中，各单位使用的网络操作系统、数据库、邮件信息系统等应用平台和业务应用系统不可能一样，各单位具体的工作程序、规则以及对信息化的需求也会存在差异，如何科学地规范检察业务应用系统的建设，在现有法律关系、工作机制和管理体制下，解决好不同应用系统间各种数据信息快速、准确、高效地进行传递、交换和共享，直接影响到检察机关网络信息系统的使用效率和水平，关系到检察机关的各项工作能否真正实现信息化、网络化，关系到能否发挥网络建设的投资效果，是检察机关信息化建设中迫切需要解决的重要问题。为了解决高检院和省级院间的信息数据交换问题，保证所有网络应用软件不论采用何种操作系统、数据库、开发平台和开发工具，都能确保数据传递、交换、检索、统计没有技术障碍，保证信息共享和业务协作的顺利实现，进而建立、完善科学、合理的检察系统网络应用软件，就有必要建立检察机关网络应用软件的数据格式规范标准，并对应用于检察行业的软件产品进行标准符合性入门测试。

再例如，作为基础应用软件的数据库管理系统，几十年来数据库技术研究非常活跃，迅速发展，与此同时，各个数据库管理系统之间的互操作性、移植性越来越受到大家的重视，标准化已经变成了数据库管理系统产品被用户接纳和认可的前提条件，各大厂商都非常注重数据库标准化工作。20 世纪 90 年代以后，数据库的应用向多元化方向发展，大型的应用往往涉及不同的应用领域，需要不同模型的数据库，再加上开发工具的发展，促进了数据库标准的成熟与发展，国产数据库系统的标准符合性测试也纳入正轨。

总体而言，标准符合性测试就是测量产品的功能和性能指标，与相关国家标准或行业标准所规定的功能和性能指标之间符合程度的测试活动。它区别于一般的测试，标准

符合性测试的测试依据和测试规程一定是国家标准或行业标准，而不是实验室自定义的或其他的有关文件。

标准符合性测试通常有两种方式，一是自律测试，二是权威测试。前者由应用开发者利用适当的工具对自己开发的产品进行测试，测试工具通常由行业协会提供，有的还可从网上免费获取或自主开发测试程序；后者则是委托授权的机构对厂商提交的产品进行测试，测试机构将出具规范化的测试报告，作为进一步产品标准化认证的技术依据，即第三方认证（从事信息技术产品认证工作的第三方合格评定组织，依据我国已发布的信息技术产品强制性标准、推荐性标准和国家有关产品认证的法律、法规和规章，代表国家对相关产品及相关设备进行认证和检测工作）。

13.2　标准符合性测试主要分类

如前所述，标准符合性测试的测试依据是我国已发布的信息技术产品强制性标准、推荐性标准等相关国家标准或行业标准，尽管不同行业依据的标准本身可能千差万别，但从内容来分，主要分为以下四类。

1．数据内容类标准

这类标准主要描述用于数据交换与互操作的数据格式或内容规范。例如中华人民共和国教育部颁发的《教育管理信息化标准》（第 1 部分《学校管理信息标准》），它包括与学校管理有关的信息集与代码集两部分。关于信息集的检测，要求验证被测信息数据表的表名称、字段名称、字段类型及长度是否均符合标准；关于代码集的检测，要求验证被测代码数据表中的代码编号、代码名称是否符合标准。

再如，检察机关网络应用软件的数据格式规范与代码符合性规范，前者定义了适用于检察机关各类业务所需的 70 多个 XML Schema，后者定义了各种基础代码的代码编号及代码含义。

2．通信协议类标准

这类标准主要描述用于数据通信与传输的接口数据格式。例如智能交通管理 NTCIP 协议，包括：NTCIP 1201 v02.26《国家运输 ITS 通信协议(NTCIP)全局对象定义》和 NTCIP 1201 v02.26《国家运输 ITS 通信协议感应式交通信号机单元对象定义》等，分别定义了交通管理中心和外场设备之间的信息传递，通过使用 NTCIP 申请层服务传达请求存取，或修正一个设备中储存的对象和感应信号机所支持对象的定义。

再例如，中国远程教育 CELTS-20 教学管理标准中的基于 HTTP 协议绑定规范。该标准描述了一个基于 HTTP 协议的远程教学管理平台，学生与教学平台进行交互操作时所需的数据形式及内容。



3．开发接口类标准

这类标准主要描述开发接口规范。如：国产数据库系统标准符合性测试包括的 SQL 标准符合性测试，ODBC 标准符合性测试、JDBC 标准符合性测试等。

- 所谓 SQL 标准符合性测试，即测试数据库管理系统与 SQL 标准的符合程度，分为语法级（即所接受的 SQL 语言与标准 BNF 的符合程度）、语义级（即数据库管理系统所实现 SQL 语言执行结果与 SQL 标准规定结果的符合程度）。目前美国标准技术研究所（NIST）的 SQL 测试用例库代表了符合性测试的主流，数据库系统对该测试用例库的支持程度是数据库管理系统标准化的重要指标。
- ODBC 规范为应用程序提供了一套高层调用接口规范和基于动态链接的运行支持环境。使用 ODBC 开发数据库应用程序时，应用程序调用的是标准的 ODBC 函数和 SQL 语句，数据库的底层操作由各个数据库的驱动程序完成。
- JDBC 规范为 Java 语言访问关系数据库提供了一个编程接口规范，它由一组用 Java 编程语言编写的类和接口组成。JDBC 规范为数据库开发人员提供了一个标准的 API，使他们能够用纯 Java API 来编写数据库应用程序。目前 JDBC 规范的最新版本为 JDBC3.0 规范。

4．信息编码类标准

如 GB18030 汉字编码标准，它是为了适应信息处理、信息交换的应用，对字符集提出了多文种、大字量、多用途的要求而定义的。GB 18030 中文符合性包括字汇完整性和体系正确性两方面。其中字汇完整性指被测软件产品的字汇范围应为国家标准 GB 18030-2000 中所有给出字形的字符，包括 27484 个汉字，总编码空间超过 150 万个码位；体系正确性要求被测软件产品对于符合 GB 18030 编码的文本文件（该文本文件可以包含单字节、双字节和四字节字符），应能够进行正确识别、处理、交换、存储、传输、显示、输入和输出。

13.3 测试策略

由于标准符合性测试的不同分类，其相应的测试原理也不尽相同。

13.3.1 数据内容类标准

如《教育管理信息化标准》（第 1 部分《学校管理信息标准》），在测试工具设计上，其实现原理如下所示。

- 将符合标准的信息集（表结构）与代码集（表内容）构建在测试工具数据库中，即建立标准模板；

- 测试工具通过 ODBC、JDBC 等数据库连接方式连接被测软件的数据库；
- 测试工具提供人工或自动方式建立模板库与被测库之间的关联，读取并验证相关数据表信息；
- 生成信息集与代码集标准符合性检测结果报告。

注意，在实际应用中，从易维护的角度出发，被测软件的代码集可能不是多个不同类别的小代码表集，而是一个包含各种类别的大代码表，但测试工具模板库往往是多个不同类别的小代码表集，这就要求测试工具能够实现一对多或多对多的关联设置。

而对于检察机关网络应用软件的数据格式规范与代码符合性规范的测试工具，可采用网上已有的相关工具或自行开发。如数据格式规范测试可辅助采用已有的 XML 解析器进行，而代码符合性规范可采取自行开发测试工具方式执行，测试步骤包括工具中建立标准模板、连接被测软件、与标准模板比对测试和输出测试结果。

13.3.2 通信协议类标准

测试工具的实现原理与第一类标准基本相似，如中国远程教育 CELTS-20 教学管理标准中的，基于 HTTP 协议绑定规范的，测试工具可以这样实现：①建立标准模拟课件；②导入模拟课件到被测平台；③ 测试工具自动运行模拟课件，主动与被测平台进行数据通信；④ 将二者通信内容与工具中的标准模板内容进行比较，得出比较分析结果。

13.3.3 开发接口类标准

1．SQL 标准符合性测试

按照 SQL92/97 标准，全面测试一个 SQL 产品的功能特性。在详细研究美国标准技术研究所（NIST）的测试用例库（即在整个测试过程中，只需要执行全部的测试用例文件，最后统计通过的测试用例即可）的基础上，可自行开发一个集测试和结果的定量分析于一体的自动化测试工具，利用该测试工具可以直接选择被测文件，运行并统计运行的结果。

通过的入门级测试用例数占入门级测试用例总数的比例，即为入门级测试通过率。通过的过渡级测试用例数占过渡级测试用例总数的比例，即为过渡测试通过率。

为了保证测试结果的真实性，还可采用交互式测试用例验证测试结果，如果发现问题，则相应的嵌入式测试用例的结果视为不通过。

2．ODBC 标准

可采用 SWsoft Inc 开发的 ODBC2.5 标准符合性测试工具进行测试。在此基础上，按照 ODBC3.0 标准对测试用例进行相当规模的修改和扩充，并且将微软的 QUICK TEST 测试工具的部分模块集成到该测试工具中，同时对测试结果进行了定量的分析。



其中，对API函数的测试，参照微软的测试工具（QuickTest）对每个函数选定一种最简单的参数组合来测试，仅用其作简单的支持性测试。此项测试根据通过测试的函数的百分比来计算。对于其他的更重要的应用功能，是通过其他更详细、更复杂的测试用例来验证的，其执行结果的成功与否直接记录为测试结果。

3．JDBC标准

可在SUN公司开发的JDBC标准符合性测试工具基础上，按照JDBC3.0标准对测试用例进行修改和扩充，同时加入对测试结果的定量分析功能。

JDBC标准符合性测试完成后，统计各个接口或类中API函数通过的测试用例点的数量，按用例通过的比例和每个类或接口所占的权值计算总体得分。

13.3.4 信息编码类标准

例如，对GB18030中文符合性测试，包括字汇完整性和体系正确性两方面。

对于字汇完整性可采用抽样测试的方法，其过程如下。

- 生成标准测试文件。即依照GB 18030的字符集生成字符数据文件（如.TXT），包括GB 18030中定义的全部汉字区、符号区、保留区和用户自定义区。
- 运行被测软件，打开已生成的标准文本文件，将屏幕显示内容与GB 18030中指定内容进行对比，记录屏幕显示对比结果。
- 运行待测软件，打开已生成的文本文件并打印其内容，将打印结果与GB 18030中指定内容进行对比，记录打印对比结果。
- 抽样对比。例如：抽样方法可定义为单字节抽样率达到100%，双字节1区抽样率达到约20%，双字节2区抽样率达到约15%，双字节3区抽样率达到约10%，双字节4区抽样率达到约5%，双字节5区抽样率达到约20%和四字节区抽样率达到约5%。抽样范围包括边界字符和中间随机字符，如有错误则抽样率加倍，直至抽样率达到100%。各区矩阵的抽样率均应达到100%。抽样对比测试办法如下：单字节区，逐字对比。双字节1～5区，以第一字节相同的所有字符构成一个矩阵为一个检查单位，每矩阵抽查第一个字符、最后一个字符，在其他字符中按前述抽样率随机抽查数个字符，如果被抽样字符中出现对比结果不符合现象，或发现明显的“？”、方框、连续空白，则按前述抽样方法进行。双字节用户区1～3，与用户文档中承诺的用户自定义字符列表或用户自定义界面的输入结果进行对比，抽样率为10%；如没有用户自定义字符，则应不显示字符。四字节区，每区抽查第一个字符、最后一个字符，在其他字符中随机抽查数个字符（区抽样率≥5%），如果被测字符中出现对比结果不符合现象，或发现明显的“？”、方框、空白，则对比整个矩阵。

对于体系正确性测试，其测试过程包括：

- 生成随机文件，即从 GB 18030 定义的全部字符中随机抽取，而形成的大于 5000 字符的文本。文本中包括单字节区、各双字节区、四字节区中的字符，所有字符随机组合。
- 编辑处理，即在被测的软件平台上，将已生成的随机文件打开，并进行编辑处理，包括插入字符、删除字符、存储字符、复制粘贴、打印等操作，各类操作均包括单字节区、各双字节区、四字节区中的字符。
- 记录结果，即记录编辑处理文本文件的结果。

对于字汇完整性，符合以下所有条件的，字汇完整性成绩为通过，其他情况为不通过。

- 单字节区显示和打印的符合率均等于 100%。
- 双字节各区显示和打印的符合率均大于 98%。
- 四字节区显示和打印的符合率均大于 97%。

对于体系正确性，插入字符、删除字符、存储字符、复制粘贴、打印等编辑操作处理正确为通过，出现乱字符、多字符、丢字符或其他影响编辑操作的处理结果为不通过。

只有在字汇完整性与体系正确性的成绩均为通过时，总成绩为通过。其他情况为不通过。

目前，由于 GB18030 的测试主要依靠人工验证，所以测试过程相对繁琐一些。

13.4 测试实施

标准符合性测试工具与一般功能和负载压力测试工具有着明显的不同，它是为明确的应用对象和测试目的服务的，具有更强的针对性，应用范围相对而言更具体、更狭隘一些。标准符合性测试的基本原理，就是将被测软件产品的功能与性能指标，和标准规定必须满足的功能和性能指标进行比较，从而确定软件产品对标准的符合程度。

一般来说，标准符合性测试可以按以下步骤实施。

① 阅读和理解标准：很多人可能不理解或者不认为应该将它归为标准符合性测试的第一步，但它确实是实施有效的标准化符合性测试的前提。因为，大多数情况下制定标准和进行标准符合性测试的不是同一组人，因此，在测试正式开始之前，首先就必须很好地阅读并理解标准的目的、意义、范围和具体的指标内容，否则测试结果就会产生偏差。

② 确定测试工具：标准符合性自动化测试一般需要依靠特定的测试工具来完成，可以选择适当的商业化测试工具，也可以根据情况决定自主开发相应的测试工具。如前所



述，由于标准具有特定性，所以大多数情况下，针对标准的符合性测试工具，需要测试组自行开发或者修改已有的测试工具。如果需要开发测试工具，则必须执行一个严格的开发流程，确保测试工具本身的正确性和有效性。

③ 确定用例文件：对于测试标准并不包含测试用例的，测试组需要根据标准规定的格式定义各种测试用例，当然应该包含正常的和异常的测试用例。

④ 执行用例文件：确定了相应的测试工具和测试用例后，就可以执行测试并记录测试执行的结果。

⑤ 分析测试结果："标准符合性"顾名思义应该就有一个测试结果基准库，通常情况下，它规定了输入与输出的对应关系，标准符合性的测试过程就是将测试用例（被测产品）的输入输出与基准库定义的输入输出相比较，从而对与标准不一致的输入输出进行统计分析，确定测试结果以及被测产品对标准的符合程度。

在测试结果的分析与评价上主要有两种形式：一种认为要全部符合标准才算通过，即 Yes or No 方式；一种则通过测试符合标准的程度来判定，如认为 80%以上的符合率即为基本符合标准。

正是信息技术的深入发展，及相关应用间方便快捷地进行通信和数据交换的迫切需要，使得信息技术标准化和标准符合性测试的重要性日益凸现。

在实际应用中，根据不同的层面，对标准的分类有多种方式，这里仅从信息技术不同标准的内容划分为主要的四类，并就相应常用的测试原理进行了阐述。

标准的价值在于应用。因此，如何准确高效地实施测试、衡量标准的应用是重要的环节。第三方测试机构代表国家对相关产品及相关设备进行评测的过程中，也是严格依据标准本身对产品进行标准符合性测试的，这必将进一步推动我国信息技术标准化建设更上新的台阶。

第 14 章　易用性测试

14.1　概述

软件是用来使用的，一个软件开发完成之后，除了要满足可用性（正确性）之外，还要有很好的易用性，最终用户是否感到软件容易使用，直接决定了一个软件能否取得市场的成功。比如 Windows 操作系统在技术上并不比 UNIX 更先进，但是 Windows 操作系统在桌面领域却风靡全球，关键在于它美观易用。而失败的例子也比比皆是，一个单位开发了办公自动化系统，配备了很好的机器设备和网络环境，领导也非常重视，但是使用了一段时间之后就把它束之高阁了，又重新回到了手工时代，问题的关键在于该系统没能很好地体现用户的使用习惯，不符合行业特点。

易用性是指软件产品被理解、学习、使用和吸引用户的能力。软件是否易用、“友好”已经成为软件质量的一个重要体现。易用性同时是一个很广泛的概念，它涉及到易理解性、易学习性、美观性，一致性、业务符合性等方面，对于测试工程师来说，易用性测试是非常富有挑战性的工作，因为易用性测试往往要依靠工程师的经验以及对行业知识的深刻理解，而对一个具有复杂业务逻辑的应用系统来说，进行易用性测试，往往还需要用户的参与。

易用性测试不仅是针对应用程序的测试，而且还要包括用户手册等系列文档，关于文档易用性测试方面的内容可以参见文档测试部分，这里主要描述的是如何对应用程序进行易用性测试。对易用性测试我们分四部分进行讨论：安装测试、功能易用性测试、界面测试和辅助系统测试。

14.2　安装测试

除了嵌入式软件之外 ，安装是软件产品实现其功能的第一步。对于一般的应用软件来说，最早体现其易用性的就是软件安装。现在的软件系统越来越庞大，有可能使安装过程变得复杂，安装耗时也会越来越长。没有正确的安装根本就谈不上正确的使用，因此安装测试就显得尤为重要，安装的易用性是安装测试的主要内容。

安装测试的方法很简单，就是按照用户安装手册安装软件，来评估安装过程的易用



性、正确性。那么对于安装测试需要注意一些什么呢，我们认为至少应该从以下几个方面来考虑。

- 安装手册的评估。在安装前需要检查安装手册或用户文档中的安装说明，一般来说，安装手册需要对安装平台、安装过程需注意的事项以及需手动配置的部分进行详细说明。
- 安装的自动化程度测试。由于制作安装程序的软件很多，其中很成熟的有Installshield 等，很多软件采用了自动安装的方式。但由于部分软件的特殊性，有时必须采用一定的手动配置来完成安装。我们要评估软件安装过程的自动化程度。一般来说，软件的安装程序尽量要做到“全自动化”，即使在不得已的情况下需要进行手动配置，也要采取一些措施，比如选择框方式等，使手动配置变得简便和明确。
- 安装选项和设置的测试。在安装过程中常常需要对安装的项目进行选择，也可能要设置不同的信息，比如安装路径等。安装测试时需要对不同的选项和设置方案进行测试，验证各种方案是否都能安装成功。
- 安装过程的中断测试。一个大型的软件有可能需要数小时来进行安装，如果因为断电、文件冲突或读写错误导致安装过程的非正常中断，有可能使已进行的安装工作前功尽弃。一个好的自动化安装程序应该能记忆安装的过程，当恢复安装时，安装程序能自动进行检测，并从“断点”继续安装。
- 安装顺序测试。对于大多数应用系统，特别是分布式系统，常常需要安装软件系统的不同组成部分。不同的安装顺序常常会导致安装失败，或者会引起一些不可预料的错误，例如，先安装客户端后安装服务器，会导致某些软件的客户端与服务器连接不上。如果《安装手册》中未明确指出安装顺序，则需要测试不同顺序的安装过程。
- 多环境安装测试。不同的应用环境下安装的情况也是不一样的，我们至少要在标准配置、最低配置和笔记本电脑三种环境中进行安装测试。很多情况下产品声称的最低配置并不符实，所以最低配置环境测试是非常必要的。另外，有些系统级的软件常常在笔记本电脑上安装时发生错误，例如，由于笔记本电脑的高集成度特性，Linux 桌面操作系统在笔记本安装时出现硬件兼容性问题。
- 安装的正确性测试。在上述的安装测试后，都需要进行简单的使用以验证安装的正确性。另外，还要考察对其他应用程序的影响。
- 修复安装测试与卸载测试。修复安装测试指软件使用后，根据需要添加或删除软件的一些组件或者修复受损的软件。修复安装和卸载也应该是自动化的，通常情况下，安装、修复安装以及卸载是一个完整安装程序中的不同选项。进行

修复安装测试时，需检查修复对软件有无不良的影响，例如，修复可能造成系统数据丢失。卸载测试重点检查卸载是否完全，不能完全卸载时有无明确提示信息等。

14.3 功能易用性测试

功能易用性的概念范围很广，这里列出了一些比较重要的功能易用性测试项，如下所示。

- 业务符合性：软件使用的目的是替代部分人工劳动，提高工作效率，因此，软件必须符合其所服务的领域的业务逻辑。这就要求软件的界面风格、表格设计、业务流程、数据加密机制等的设计必须符合相关的法律法规、业界标准规范以及使用人员的习惯。
- 功能定制性：为了适应用户需求的不断变化，软件功能应当能够灵活定制，如电子政务软件的公文流转节点，应可以灵活定义；工资软件中部门结构和人员归属应可灵活调整等。
- 业务模块的集成度：在一个系统中业务模块之间有可能存在较紧密的关联，例如在 ERP 系统中，采购某些零部件之后必须进行质检，这样的业务需求造成"采购管理"模块与"质量检测"模块存在直接的关联，那么用户能否在"采购管理"用户操作界面下，直接进入"质量检测"模块，并且"采购管理"模块中的零部件数据能否直接传递给"质量检测"模块。
- 数据共享能力："一次输入、多处应用"不仅能够减少用户的重复输入工作，更有效地保证了数据的正确性。在软件设计中必须充分考虑数据库表的关联和数据重用问题，最大程度地减少用户的重复输入，同时保证数据传递的一致性。
- 约束性：对于流程性比较强的业务操作，上一步操作完成之后，要强制进行下一步操作，这时需要软件以向导或与屏蔽无关操作的方式来限制用户的操作；另外，应以屏蔽或提示的方式阻止用户输入非法字符或进行损害数据和系统的操作，这样才能有效地避免用户犯错误，同时也减少了系统出现异常的概率，提高系统的安全可靠性。
- 交互性：包括用户操作的可见性和系统对用户的反馈。对于用户的每一步操作都应有所回应或者提示，使用户清晰地看到系统的运行状态。例如，在执行复制操作时，至少应该向用户反馈操作持续时间，显示计算机正在工作，没有停滞或者报错。对于用户来说，这种回应与提示是对用户操作的认可与尊重，更有助于用户确定下一步操作该如何进行。



- 错误提示：关键操作完成后或数据删除等操作前给出明确提示，操作错误或系统出现错误时，给出的出错信息中提供差错产生的原因，并指示如何进入正确的步骤，帮助用户从错误中恢复。

14.4 用户界面测试

用户界面测试主要核实用户与软件之间的交互，验证用户界面中的对象是否按照预期的方式运行，并符合国家或行业的标准。

界面测试中的部分工作主观性比较强，测试结果往往与测试人员的喜好有关。因此，界面测试的一个缺点就是，测试人员在整个测试过程中身心不可能保持一致，在一定程度上会影响测试结果的准确性。

用户界面测试可分为整体界面测试和界面中的元素测试。界面中的元素主要包括窗口、菜单、图标、文字、鼠标等。

14.4.1 界面整体测试

界面整体测试是指对界面的规范性、一致性、合理性等进行测试和评估。

1．规范性测试

软件的界面要尽量符合现行标准和规范，并在应用软件中保持一致。而开发软件时就要充分考虑软件界面的规范性，最好的办法是采取一套行业标准。现在许多行业已有自己的标准，如 IBM 标准、Microsoft 标准、Apple 标准。这些标准已经基本包含“菜单条、工具栏、工具箱、状态栏、滚动条、右键快捷菜单”的标准格式，以上标准已经基本完善，对某些行业只需利用已有的成果就可以。

对于一些特殊行业，由于系统使用环境和用户使用习惯的特殊性，使用以上标准是远远不够的，还要对自身特殊的需要加以补充。如软件的用户定位包括不同年龄阶段的用户，那么就会有一些用户基本不使用鼠标右键，年龄较大的用户难以看清密集的较小的文字，或用户对计算机系统和网络不够熟悉，或硬件环境一般，甚至比较差，少有配置优良的计算机等等。在这种环境下，用户对计算机的使用一般没有使用倾向，大多更适应手工操作。像这种情况，特殊行业都要有一套自己比较完善的标准和规范。

这些标准和规范是经过各种类型的测试与评估，不断总结，经验积累和反反复复设计成果。在界面测试中，测试工程师应该严格遵循这些标准和规范设计界面规范性测试用例。

2．合理性测试

界面的合理性是指界面是否与软件功能相融洽，界面的颜色和布局是否协调等。如

果界面不能体现软件的功能，那么界面的作用将大打折扣。所以，界面的合理性是界面美的首要因素，它提醒设计者不要片面追求外观漂亮而导致失真或华而不实。

合理性差的界面无疑会混淆软件意图，致使用户产生误解。即使它不损害软件功能与性能，也会使用户产生不该有的情绪波动。合理的用户界面是应用程序的一个重要组成部分，也是使软件易用的重要基础。空间使用应当形成一种简洁、有序、易于操作的布局，使信息组织具有艺术性。如果一个界面上有太多或者杂乱无章的控件，会给用户寻找字段或者控件带来不便和困难。

测试软件界面的合理性一般通过观察进行，举例如下。

- 界面中元素的文字、颜色等信息是否与功能不一致；
- 前景与背景色搭配是否合理协调，反差是不是太大；
- 界面中的元素大小和布局是否协调；
- 窗口的比例是否合适。

3．一致性测试

一致性既包括使用标准的控件，也指相同的信息表现方法，如在字体、标签风格、颜色、术语、显示错误信息等方面确保一致。好的软件界面都具有相似的界面外观、布局、交互方式以及信息显示等。界面保持高度一致性，用户可以减少过多的学习和记忆量，从而降低培训和支持成本。

此外，软件的界面在不同平台上是否表现一致呢？作为测试人员不要忽略这一点。如颜色、字体，有时有些软件在不同的平台表现得不尽如人意“——”在一个系统上看上去很好，在另一个系统上常常看上去很糟。

对于在不同的平台测试软件界面的一致性可以用下面的方法：在不同分辨率下，观察界面的美观程度，分别在 800×600，1024×768，1152×864，1280×768，1280×1024，1200×1600 大小的字体下进行测试。一个好的软件要有一个默认的分辨率，而在其他分辨率下也都能运行。

作为测试工程师，在测试界面一致性时应该注意以下几点因素。

- 布局是否一致，如所有窗口按钮的位置和对齐方式要一致；
- 标签和讯息的措辞是否一致，如在提示、菜单和帮助中产生相同的术语；
- 界面外观是否一致，如控件的大小、颜色、背景和显示信息等属性要一致，但一些需要艺术处理或有特殊要求的地方除外；
- 操作方法是否一致，如双击其中的项，使得某些事件发生，那么双击任何其他列表框中的项，都应该有同样的事件发生；
- 颜色的使用是否一致，颜色的前后一致会使整个应用软件有同样的观感；
- 快捷键在各个配置项上语义是否保持一致。



表 14-1 列出了常用的快捷键及其功能。

表 14-1 常用的快捷键及功能

功　能	快 捷 键	功　能	快 捷 键
删除	Ctrl-D	粘贴	Ctrl-V
寻找	Ctrl-F	关闭	Ctrl-W
拷贝	Ctrl-C	剪切	Ctrl-X
替换	Ctrl-H	打印	Ctrl-P
插入	Ctrl-I	新记录	Ctrl-N
打开	Ctrl-O	保存	Ctrl-S
缺省按钮确认操作	Enter	取消按钮、取消操作	Esc
MS Windows 保留键			
下一窗口	Ctrl-Tab	任务列表	Ctrl-Esc
关闭窗口	Ctrl-F4	结束应用	Alt-F4
下一应用	Alt-Tab	上下文相关帮助	Shift-F1

4. 界面定制性测试

对于适用于多层次用户的软件，由于用户熟练程度（外行、初学、熟练）不同、使用频度不同和不同角色，需要不同的操作方式或用户界面。如财务软件中财务总监的界面应提供大量查询功能和更多使用鼠标的操作，而会计、出纳的界面应提供更多的键盘快捷方式和以最少的步骤完成日常凭证制作审核。因此需要对界面的可定制性进行测试，测试中可参考以下几项测试内容。

- 界面元素的可定制性。可以允许用户定义工具栏、状态栏是否显示，工具栏显示在界面上的位置，如上方、下方或悬浮等；一些软件还可以定义菜单的位置。随着 Windows XP 的出现，界面风格的可定制也为软件个性化提供了新的特性。
- 工具栏的可定制性。工具栏为用户使用常用的功能提供了方便，但不同用户对“常用”的理解是不同的，因此，应当允许用户自定义工具栏，包括建立新的工具栏，选择要显示的工具栏，定义工具栏上的按钮，制定为工具按钮定义所链接的功能等。
- 统计检索的可定制性。检索和统计是用户向系统索取数据最经常用到的功能，检索条件是否灵活、分类统计是否合理、是否允许用户定义检索条件和统计项，需要测试人员在充分了解用户需求和使用习惯的基础上，制定大量案例，通过实际操作来体会。

- 报表的可定制性。各种各样的报表是软件对用户输出的重要方式，报表表头包括的项目、表格的行高列宽、表中数据的单位和显示格式超长超宽表的分页方式等如果能够允许用户自定义，则可以使软件生成的报表适用于更广泛的范围，减少用户二次处理表格的工作量，极大地方便用户的使用。

14.4.2　界面元素测试

1．窗口测试

在 Windows 平台上运行的应用软件，窗口是软件界面的基础，正如操作系统的名字。窗口是显示设备中的一个区域，用于观看对象、对象相关信息以及应用与对象的动作进行交互。窗口有标题栏可以进行打开、关闭、创建、缩放、移动、删除、重叠等操作。

现在让我们了解一下窗口的基本组成部分，从外观上讲，一般窗口是由标题、边框、菜单、工作区、滚动条等组成（如图 14-1 所示）。虽然软件产品的窗口各种各样，令人眼花缭乱，但基本组成是相同的。

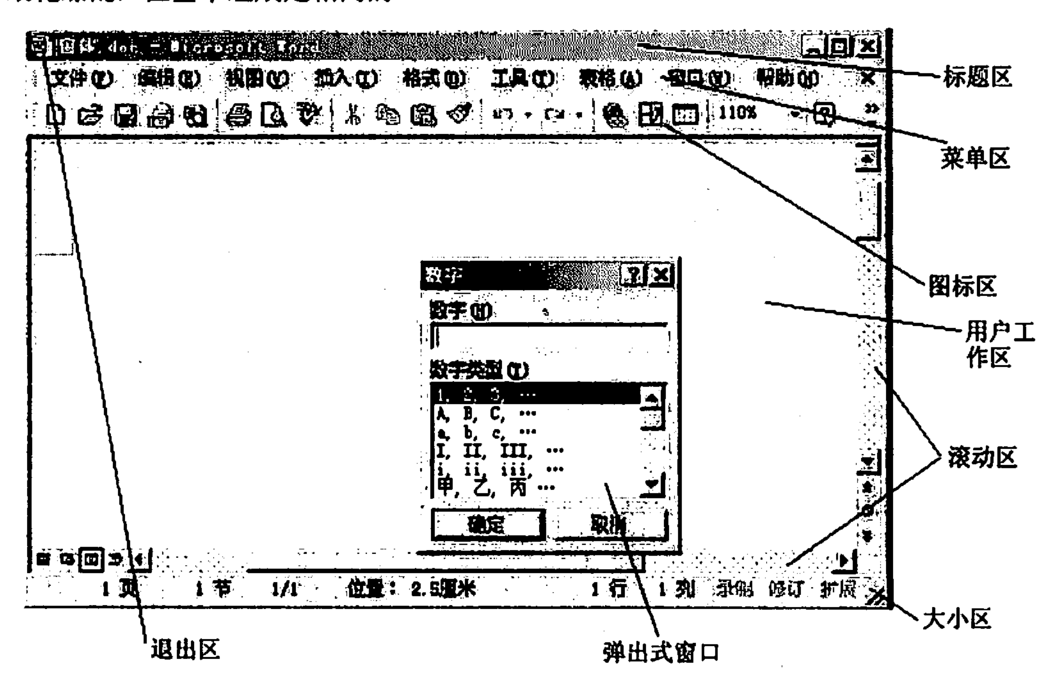


图 14-1　Microsoft Word 窗口的配置部件示意图

下面是一些窗口测试用例设计的参考例子。



- 窗口控件的大小、对齐方向、颜色、背景等属性的设置值是否和程序设计规约相一致。
- 是否显示相关的下拉菜单、工具条、滚动条、对话框、按钮、图标和其他控制，既能正确显示又能调用。
- 若窗口无法显示，所有内容是否能够改变大小、移动和滚动。
- 活动窗口是否能够被反显加亮。
- 窗口是否正确地关闭。
- 多个窗口叠加时窗口的名称是否显示正确。
- 窗口的数据是否能够利用鼠标、功能键、方向箭头和键盘操作。
- 当窗口被覆盖并重新调用后，窗口是否能够正确再生。
- 如果使用多任务，是否所有的窗口被实时更新。
- 窗口是否支持最小化和最大化或放大。
- 窗口上的控件是否随着窗体的缩放而缩放。
- 父窗体支持缩放时，子窗体是否也缩放。
- 在一个窗口中按 Tab 键，移动聚焦是否按顺序移动。Tab 的顺序应是先从上至下，再从左至右。
- 子窗口位置是否在父窗口的左上角或正中，由于屏幕对角线相交的位置是用户直视的地方，正上方 1/4 处为易吸引用户注意力的位置，在放置窗口时要注意利用这两个位置。父窗口或主窗口的中心位置应该在对角线焦点附近。
- 当多个子窗口弹出时是否依次向右下方偏移，以显示出窗口标题为宜，如图 14-2 所示为 Microsoft PowerPoint 窗口重叠示意图。通常重叠的窗口具有固定大小和位置，新打开的窗口要堆叠在最近打开的窗口上，这些重叠的窗口都带有突出的标签以便选择。
- 重要的命令按钮与使用较频繁的按钮是否放在了界面上醒目的位置。因错误使用而引起界面退出或关闭的按钮，放在容易点击的位置。横排开头或结尾，与竖排结尾为容易点击的位置。
- 与正在进行的操作无关的按钮应该加以屏蔽（Windows 中用灰色显示，没法使用该按钮）。
- 按钮的大小要与界面的大小和空间是否协调。避免在空旷的界面上放置很大的按钮。放置完控件后界面不应有很大的空缺位置。
- 多窗口的切换响应时间是否过长。如果切换时间过长就会使用户出现意外的焦躁情绪，而响应时间过短有时会造成用户操作节奏加快，从而导致用户操作错误。

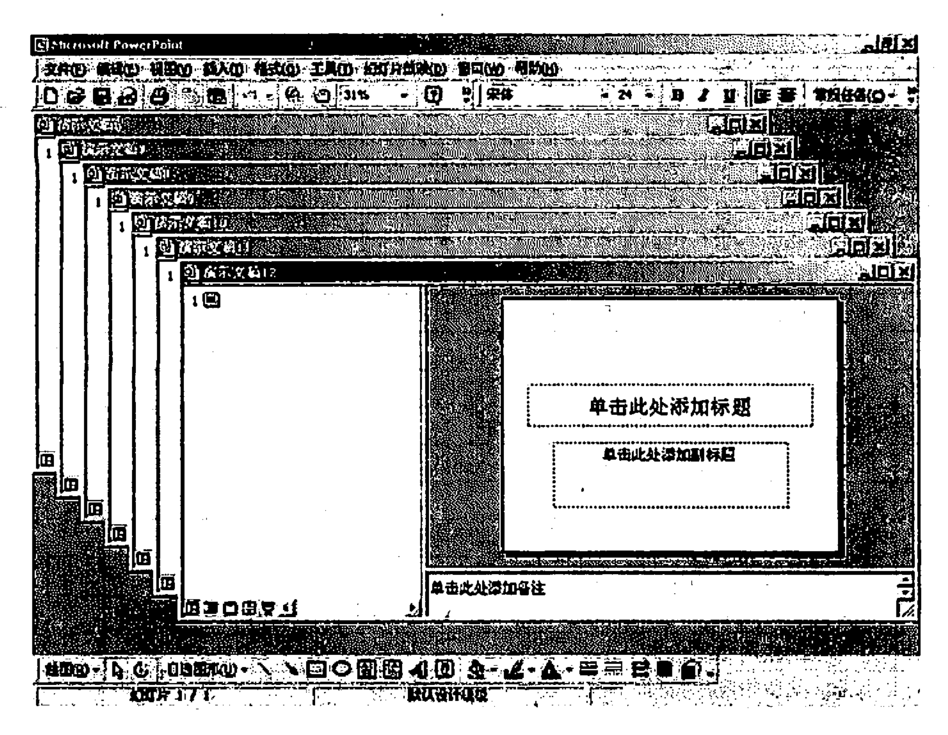


图 14-2　Microsoft PowerPoint 窗口重叠示意图

2．菜单测试

菜单对我们来讲是很熟悉的，它是应用程序命令项列表，菜单位置按照功能来组织。菜单按图形方式可以产生丰富多彩的菜单形式，例如条形菜单、水平和垂直的弹出式菜单、下拉菜单、T 形菜单等。无论采用哪种方式，仅仅是菜单显示方式不同罢了，菜单的测试方法还是基本一样的。

菜单是否易用主要体现在它能否提供线索帮助用户识别，而不用强迫用户去记忆。如果用户只通过简单的培训或偶尔的使用，就可以接受该系统，那么简单和有规则的菜单尤其有效。

作为测试工程师，设计菜单界面测试用例主要应从以下几点考虑。

- 是否符合需求；
- 菜单项的措辞是否准确；
- 菜单项的顺序是否合理；
- 图形的布局是否一致。

菜单界面测试用例设计范例如表 14-2 所示。



表 14-2 菜单界面测试用例

编号	测试案例及说明	测试结果	缺陷原因
1	菜单功能是否正确执行		
2	下拉菜单是否根据菜单选项的含义进行分组		
3	菜单是否有快捷命令方式		
4	文本字体、大小和格式是否正确		
5	菜单功能是否随当前的窗口操作加亮或变灰		
6	菜单功能的名字是否具有自解释性		
7	菜单项是否有帮助		
8	右键快捷菜单是否采用与菜单相同的准则		
9	是否可以通过鼠标访问所有的菜单功能		
10	是否适当地列出了所有的菜单功能和下拉式子功能		
11	下拉式操作能否正常工作		
12	是否根据系统功能进行合理分类，将选项进行分组		
13	菜单深度是否控制在三层以内		
14	菜单标题是否简明、有意义		
15	是否依据使用频度排列		
16	是否依据逻辑顺序排列		
17	是否依据使用顺序排列		
18	各级菜单显示格式和操作方式是否一致		

这个测试用例可以适用不同的菜单，表 14-2 是针对菜单界面测试所设计的一个测试用例说明。这个测试用例通过测试人员测试，会找出更多的 Bugs。

3. 图标测试

图标实际上属于菜单交互方式，只是它使用图标来代表文本菜单的菜单项。使用图标可以形象、逼真地反映菜单的功能，从而使理解、学习和操作变得更加易用。

由于图标是表示实体信息的简洁、抽象的符号，所以在日常生活中被广泛地使用。图标不仅仅作为表示实体的符号，还可以作为可视按钮项，当被选中激活时，可以完成指定的功能。

图标测试比较主观，与测试人员的喜好有关。比如，图标基调颜色刺眼，用户登入界面比较难于找到，图标比较抽象，图标范围太广等都属于用户界面测试中的缺陷。

形象的图标给人很大的帮助，使人容易理解其内涵。那么图标测试用例要考虑的重点有哪些呢，以下所提供的几点可以作为参考。

- 图标是否符合常规的表达习惯。
- 不同的目标是否采用不同的图标。

- 图标是否具有清晰的轮廓，轮廓清晰的图标可保证图像在不同背景色上都具有较好效果。
- 注意图标的尺寸，建议图标的尺寸小一些较好。如工具栏图标非常小，您使用简单的图像，以直观方式显示图像即可清晰地表达图标的含义，而不必使用其他的复杂方式。Windows XP 图标有四种尺寸，建议使用以下四种尺寸：48×48 像素，32×32 像素，24×24 像素以及 16×16 像素。
- 建议图标的外形与实际功能相似，应尽量避免抽象。这样的图标可以使用户很轻松、容易地认识此图标。
- 在图标上是否加有标注。

4．鼠标测试

鼠标问题经常被人们忽略，但我们无时无刻都不能离开它。用户会把鼠标移进、移出窗口，或当光标在窗口中，用户按下、释放鼠标键，鼠标是否准确、灵活，对一个测试人员来说必将提到日程上来。

以下所提供的几点可以作为鼠标测试的参考。

- 在整个交互式语境中，是否可以识别鼠标操作；
- 如果要求多次点击鼠标，是否能够在语境中正确识别；
- 如果鼠标有多个按钮，是否能够在语境中正确识别；
- 光标、处理指示器和识别指针是否随操作恰当地改变；
- 点击选中而不是滑动停留选中；
- 支持滑轮（鼠标中间的滚动轮）上下翻动操作；
- 对于相同种类的元素采用相同的操作激活；
- 用沙漏表示系统忙，用手型表示可以点击；
- 鼠标无规则点击时是否会产生无法预料的结果；
- 单击鼠标右键是否弹出菜单，取消右键是否隐藏弹出的菜单。

5．文字测试

文字在视觉上向用户传达作者的意图和各种信息，如果文字的组合巧妙，在视觉传达的过程中能够给人以美的感受，从而获得良好的心理反应。反之，则使人看后心里不愉快，视觉上难以产生美感，甚至会让用户拒而不看，这样势必难以传达出作者想表现的意图和构想。

要达到这一目的必须考虑文字的整体诉求效果，给人以清晰的视觉印象。因此，在测试过程中，测试人员应该注意文字是否繁杂零乱，使人易认、易懂，测试文字主要依靠软件设计标准，观察文字是否有效地传达作者的意图，表达设计的主题和构想意念。

文字测试是测试软件中是否拼写正确，是否易懂，不存在二义性，没有语法错误；



文字与内容是否有出入等等，包括图片文字。比如，“请输入正确的证件号码”中何谓正确的证件号码。证件可以为身份证、驾驶证，也可为军官证，如果改为“请输入正确的身份证号码”用户就比较容易理解了。

14.4.3 界面测试典型用例

现在介绍一个菜单界面测试的模拟，软件人员开发了第一版的软件，如图 14-3 所示为菜单测试用例，供测试人员测试，此时测试人员将根据上面章节中提到的测试用例来找出软件的问题。

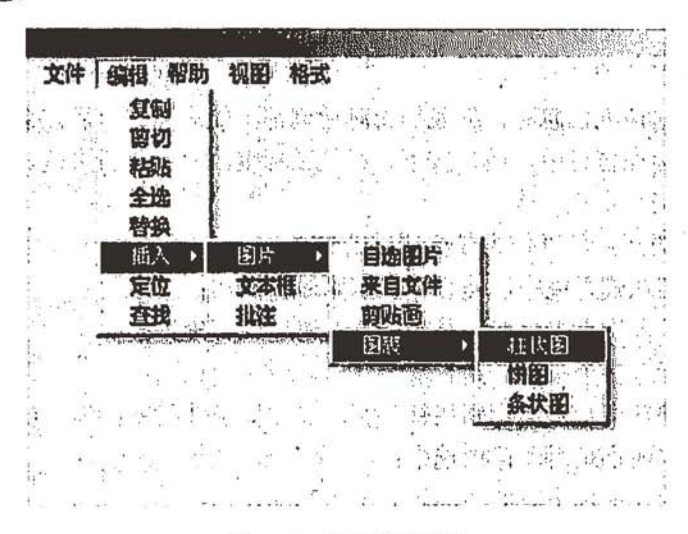


图 14-3 菜单测试用例

菜单界面测试用例设计范例如表 14-2 所示。

这个测试用例可以适用不同的菜单，如表 14-3 所示是针对菜单界面测试所设计的一个例子。这个测试用例通过测试人员测试，会找出更多的缺陷。

表 14-3 菜单界面用例测试用例

编号	测试案例及说明	测试结果	缺陷原因
1	菜单功能是否正确执行	通过	
2	下拉菜单是否根据菜单选项的含义进行分组	通过	
3	对菜单是否有快捷命令方式	不通过	为菜单提供更多的选择途径，应为菜单选择提供快捷键
4	文本字体、大小和格式是否正确	通过	

续表

编号	测试案例及说明	测试结果	缺陷原因
5	菜单功能是否随当前的窗口操作加亮或变灰	通过	
6	菜单功能的名字是否具有自解释性	通过	
7	菜单项是否有帮助	通过	
8	右键快捷菜单是否采用与菜单相同的准则	不通过	单击右键没有提供快捷菜单
9	是否可以通过鼠标访问所有的菜单功能	通过	
10	是否适当地列出了所有的菜单功能和下拉式子功能	通过	
11	下拉式操作能正常工作吗	通过	
12	是否根据系统功能进行合理分类，将选项进行分组	不通过	不应把插入功能键放入编辑菜单里，应把插入单做主菜单
13	菜单深度是否控制在三层以内	不通过	建议编辑菜单应控制到三层以内
14	菜单标题是否简明、有意义	通过	
15	是否依据使用频度排列	通过	
16	是否依据逻辑顺序排列	通过	
17	是否依据使用顺序排列	通过	
18	各级菜单显示格式和操作方式是否一致	通过	

14.5　辅助系统测试

辅助系统是指为了帮助和引导用户使用软件而存在于软件内的辅助性系统。辅助系统是否完整好用是软件易用性的重要体现，一般来说辅助系统包括帮助、向导和信息提示等。

14.5.1　帮助测试

软件应该提供所有规格说明和各种操作命令用法的帮助系统，使用户在使用中遇到困难时可以自己寻求解决方法。

很多用户在使用软件时常常会发现帮助系统没有及时更新，可能软件版本已经更改过数次，但配套的软件帮助说明并没有及时更新。这是开发人员和测试人员容易忽略的问题。可能开发人员和测试人员对这套系统很熟悉，并不需要更多的帮助，但用户却不是这样，当遇到操作问题或术语问题时，首先想到的就是从软件系统得到帮助。

对帮助系统的测试一般从以下几方面入手。

- 前后一致性。
- 内容完整性。
- 可理解性。
- 方便性。

如表 14-4 所示是一组帮助系统测试用例，可以作为参考。

表 14-4 帮助系统测试用例

编号	测试案例及说明	测试结果	缺陷原因
1	系统是否提供 F1 及时帮助功能		
2	在界面上调用帮助时应该能够及时定位到与该操作相对的帮助位置		
3	对功能采用及时帮助是否能准确定位到帮助系统的位置		
4	利用帮助索引是否能定位到帮助主题和内容		
5	是否具有打印功能		
6	目录是否划分有层次		
7	帮助内容描述得是否准确，一定详细到可以解决问题		
8	在系统不同的位置激活帮助内容与当前操作内容是否相关联		
9	微帮助提供：由状态栏提供或在控件上有提示文本		

14.5.2 向导测试

提到软件的易用性就不得不提软件的向导，对于应用中某些部分的处理流程是固定的，用户必须按照指定的顺序输入关键字作为信息，为了使用户得到必要的引导使用向导，使用户可以直接去找自己要去的地方，而不必像走迷宫一样乱走一通。另外，新用户在系统中可能会迷失方向，系统的向导可以引导用户怎样操作。

在测试过程中需要验证向导是否正确，确认向导的连接是否确实存在，是否每一步都有向导说明，向导是否一致，向导是否直观。最后要注意一点是向导必须用在固定处理流程中，并且处理流程应该不少于 3 个处理步骤。

14.5.3 信息提示

信息提示是计算机用信息的形式对用户的某些操作所做的反应。在一些操作中，如果系统没有反馈显示信息，用户就无法判断他的操作是否为计算机所接受，是否正确，以及操作的效果是什么。信息提示可采用多种方式：文本、图形和声音等。

即使系统具有信息提示，但如果显示信息不完整、不明确或不够智能，也无法真正

满足用户的需求。如何评测向用户提供的视觉和听觉上的反馈，确保在用户和界面之间建立双向通信是十分重要的。以下准则集中体现了如何测试信息提示。

- 提示信息是否用具有可以理解性的语言进行描述。提示信息应不依赖于外界的信息源就能一目了然。出错信息应该有明确的意义，并伴随听觉和视觉效果，如特殊的图像、颜色或信息闪烁。这类信息除报错和警告之外，还应向用户提供如何从错误中恢复的建设性意见，和指明错误的潜在危害。
- 对重要的、有破坏性的命令是否提供确认措施，以避免破坏性的操作。例如，用户请求删除文件，或表示要覆盖某些信息，或要求终止一个程序。
- 信息是否具有判断色彩，注意，任何情况下信息提示只能是引导和帮助用户，而不是指责用户。
- 信息提示是否具有统一的标记、标准的缩写和隐含的颜色。如中英文混杂，拼错单词，风格不一等经常会影响用户的理解。



第 15 章　可靠性测试

15.1　软件可靠性与可靠性测试

15.1.1　软件可靠性概述

在现代军事和商用系统中，以软件为核心的产品得到了广泛的应用。随着系统中软件成分的不断增加，使得系统对软件的依赖性越来越强，对软件可靠性的要求也越来越高。目前硬件可靠性测试技术和评估手段日趋成熟，硬件可靠性评估模型经过长期的实践积累，已经得到了业界的认可。但是由于软件和硬件存在着巨大的差异性，硬件的可靠性测试和评估技术，并不能完全应用于对软件的可靠性的测试和评估中。因此软件可靠性技术研究成为当今可靠性工程研究领域中一个新的领域。

国外从 20 世纪 60 年代后期开始加强对软件可靠性的研究工作，经过 20 年左右的研究，推出了各种可靠性模型和预测方法，于 1990 年前后形成了较为系统的软件可靠性工程体系。同时，从 20 世纪 80 年代中期开始，西方各主要工业强国均确立了专门的研究计划和课题，如英国的 AIVEY（软件可靠性和度量标准）计划、欧洲的 ESPRIT（欧洲信息技术研究与发展战略）计划、SPMMS（软件生产和维护管理保障）课题、Eureka（尤里卡）计划等。每年，都有大量的人力物力投入到软件可靠性研究项目中，并取得了一定的成果。

国内对于软件可靠性的研究工作起步较晚，在软件可靠性量化理论、度量标准（指标体系）、建模技术、设计方法、测试技术等方面与国外差距较大。

目前，软件可靠性管理方面还没有建立起具有权威性的管理体系和规范。比如，如何描述软件可靠性，如何测试、如何评估、如何设计、如何提高等。由于目前国内外对于软件可靠性模型的研究多集中在软件的开发阶段，而很少有涉及测试与评估阶段的可靠性模型，即使现有的模型，也多来源于硬件可靠性评估，与软件可靠性评估存在较大的差距，所以从事软件可靠性测试与评估研究是一个有理论价值和实际意义工作。总的来说，软件可靠性工程研究虽然得到了普遍的重视，但仍然不是很成熟，还处于发展确立阶段。

15.1.2　软件可靠性的定义

可靠性（reliability）是指产品在规定的条件下和规定的时间内完成规定功能的能力。

按照产品可靠性的形成，可靠性可分为固有可靠性和使用可靠性。固有可靠性是通过设计、制造赋予产品的可靠性；使用可靠性既受设计、制造的影响，又受使用条件的影响。一般使用可靠性总低于固有可靠性。

软件与硬件有很多不同点，但从可靠性的角度来看，它们主要有 4 个不同点：

- 复杂性。

软件内部逻辑高度复杂，硬件则相对简单，这就在很大程度上决定了设计错误是导致软件失效的主要原因，而导致硬件失效的可能性则很小。

- 物理退化。

软件不存在物理退化现象，硬件失效则主要由于物理退化所致。这就决定了软件正确性与软件可靠性密切相关，一个正确的软件任何时刻均可靠。然而，一个正确的硬件元器件或系统，则可能在某个时刻失效。

- 惟一性。

软件是惟一的，软件拷贝不改变软件本身，而任何两个硬件不可能绝对相同。这就是为什么概率方法在硬件可靠性领域取得巨大成功，而在软件可靠性领域不令人满意的原因。

- 版本更新较快。

硬件通常更新周期较慢，硬件产品一旦定型一般就不会更改，而软件产品通常受需求的变更，软件缺陷的修复，造成软件版本更新较快，这也给软件可靠性评估带来较大难度。

尽管这样，软件仍然是一种具有特殊属性的产品，因此，我们也可以按照上面的产品可靠性定义来框架性地描述软件的可靠性。

1983 美国 IEEE 计算机学会对“软件可靠性”做出了更为明确的定义，随后，此定义经美国标准化研究所批准为美国的国家标准。在 1989 年我国国家标准 GB/T-11457 也采用了这个定义。这个定义就是：

- 在规定的条件下，在规定的时间内，软件不引起系统失效的概率，该概率是系统输入和系统使用的函数，也是软件中存在的缺陷的函数；系统输入将确定是否会遇到已存在的缺陷（如果缺陷存在的话）。
- 在规定的时间周期内，在所述条件下程序执行所要求的功能的能力。

显而易见，美国 IEEE 计算机学会关于“软件可靠性”的定义仍然沿用了“产品可靠性”的定义，但有了更具体的定位和更深入的描述。

我们来分析一下软件可靠性的框架性定义。

- 规定的时间。

软件可靠性只是体现在其运行阶段，所以将“运行时间”作为“规定的时间”的度

量。“运行时间”包括软件系统运行后工作与挂起（开启但空闲）的累计时间。由于软件运行的环境与程序路径选取的随机性，软件的失效为随机事件，所以运行时间属于随机变量。

- 规定的条件。

规定的条件主要指软件的运行环境。它涉及软件系统运行时所需的各种支持要素，如支持硬件平台（服务器、台式机、网络平台等）、操作系统、数据库管理系统、中间件以及其他支持软件、输入数据格式和范围以及操作规程等。不同的环境条件下软件的可靠性是不同的，具体地说，规定的环境条件主要是描述软件系统运行时计算机的配置情况以及对输入数据的要求，并假定其他一切因素都是理想的。有了明确规定的环境条件，还可以有效地判断软件失效的责任在用户方还是开发方。

- 所要求的功能。

软件可靠性还与规定的任务和功能有关。由于要完成的任务不同，软件的运行情况会有所区别，则调用的子模块就不同（包括程序选择路径不同），其可靠性也就可能不同。所以要准确度量软件系统的可靠性，必须首先明确它的任务和功能。

- “软件可靠性”定义具有以下特点。

① 用内在的“缺陷”和外在的“失效”的关系来描述可靠性，更能深刻地体现软件的本质特点。

② 定义使人们对软件可靠性进行量化评估成为可能。对于软件的可靠性这样一个质量特性，很难用一个明确直观的数值去体现。而依据这个定义，我们有可能通过分析影响可靠性的因素，用函数的形式，按照不同的目的建立各种数学模型去分析软件可靠性。

③ 用概率的方法去描述可靠性是比较科学的。前面讲到，软件失效是随机的外部表现，完全是一个随机事件，而软件缺陷是软件固有的没有损耗的内在特质。定义用规定时间内一定的操作不出现软件失效的概率，也就是输入未碰到软件缺陷的概率，来描述可靠性，这种方法就是用概率来描述纯粹的随机事件，是比较合理的，也是可行的。

15.1.3 软件可靠性的定量描述

前一节从软件可靠性的定义我们可以看到，软件的可靠性可以基于使用条件、规定时间、系统输入、系统使用和软件缺陷等变量构建的数学表达式。下面我们从可靠性的定义中的术语“规定时间”、“失效概率”开始，探讨软件可靠性的定量描述，并相应地引入一些概念。

1．规定时间

首先对于“规定时间”我们有三种概念，一种是自然时间，也就是日历时间，指我

们日常计时用的年、月、周、日等自然流逝的时间段；一种是运行时间，指软件从启动开始，到运行结束的时间段；最后一种是执行时间，指软件运行过程中，中央处理器（CPU）执行程序指令所用的时间总和。

例如某单位有一套供会计人员使用的财务软件，我们来关注一整天的时间，上午 9:00 上班开机运行，下午 5:00 下班退出程序。在这里，自然时间是一天，也就是 24 小时，运行时间是 8 个小时，而 CPU 处理程序的执行时间可能不到 2 小时，这要视会计的业务繁忙状况、使用软件的频度和软件本身的设计而定。

很明显，在这三种时间中，我们度量软件的可靠性，使用执行时间最为准确，效果也最好。如果运行的软件系统处于一种相对稳定的工作状态，我们可以根据一定的经验值，按一定的换算比例，对这三种时间进行折算。

2．失效概率

我们把软件从运行开始，到某一时刻 t 为止，出现失效的概率看作关于软件运行时间的一个随机函数，用 $F(t)$表示。根据我们对软件可靠性的分析，函数 $F(t)$有如下特征：

- $F(0)=0$，即软件运行初始时刻失效概率为 0。
- $F(t)$在时间域（0, +∞）上是单调递增的。
- $F(+\infty)=1$，即失效概率在运行时间不断增长时趋向于 1，这也和"任何软件都存在缺陷"的思想相吻合。

为了简化分析，我们把 $F(t)$看作关于时间 t 的一个连续函数，并且可导。

3．可靠度

我们用来表示可靠性最为直接的方式就是可靠度，根据可靠性的定义，可靠度就是软件系统在规定的条件下，规定的时间内不发生失效的概率，如果用 $F(t)$来表示到 t 时刻止，软件不出现失效的概率，则可靠度的公式为

$$R(t)=1-F(t) \tag{15-1}$$

同样，我们知道 $R(0)=1$，$R(+\infty)=0$

4．失效强度

失效强度（Failure Intensity）的物理解释就是单位时间软件系统出现失效的概率。在 t 时刻到 $t+\Delta t$ 时刻之间软件系统出现失效的平均概率为$(F(t+\Delta t)-F(t))/\Delta t$，当$\Delta t$ 趋于很小时，就表现为 t 时刻的失效强度。用 $f(t)$表示失效强度函数，则

$$f(t)=\lim_{\Delta t\to 0}\frac{F(t+\Delta t)}{\Delta t}=F'(t) \tag{15-2}$$

5．失效率

失效率（Failure Rate）又称风险函数（Hazard Function），也可以称为条件失效强度，



物理解释就是在运行至此软件系统未出现失效的情况下，单位时间软件系统出现失效的概率。具体用数学用语来描述，就是当软件在 $0\sim t$ 时刻内没有发生失效的条件下，t 时刻软件系统的失效强度，用 $\lambda(t)$ 表示失效率，则

$$f(t)=\lambda(t)\cdot R(t) \tag{15-3}$$

代入公式（15-1）可得从可靠度到失效率的转换表达式：

$$\begin{aligned}\lambda(t)&=\frac{f(t)}{R(t)}\\&=\frac{F'(t)}{R(t)}\\&=-\frac{R'(t)}{R(t)}\end{aligned} \tag{15-4}$$

6．可靠度与失效率之间的换算

我们知道，在 0 时刻，可靠度 $R(0)$ 为 1，对公式（15-4）一阶常微分方程求解可得：

$$R(t)=e^{-\int_0^t\lambda(s)\mathrm{d}s} \tag{15-5}$$

假设软件系统的失效率为常数时，由公式 15-5 可得：

$$R(t)=e^{-\lambda t} \tag{15-6}$$

当失效率 $\lambda(t)$ 与时间 t 之积，也就是 $t\lambda(t)<0.05$ 时，公式 15-6 可简化为

$$R(t)\approx 1-\lambda(t)\cdot t \tag{15-7}$$

这样计算误差在 2.5%之内。

由公式 15-6 可得从可靠度到失效强度的转换公式

$$\lambda(t)=-\frac{\ln[R(t)]}{t} \tag{15-8}$$

当可靠度 $R(t)$ 大于 0.95 时，公式 15-6 可简化为

$$\lambda(t)=\frac{1-R(t)}{t} \tag{15-9}$$

这样计算误差在 2.5%之内。

7．平均无失效时间

平均无失效时间（MTTF）（Mean Time to Failure）就是软件运行后，到下一次出现失效的平均时间。通常平均无失效时间更能直观地表明一个软件的可靠程度。用 θ 表示平均无失效时间 MTTF，则可得：

$$\theta=\int_0^{+\infty}R(t)\mathrm{d}t \tag{15-10}$$

代入关于失效率的换算公式，可得

$$\theta = \int_0^{+\infty} e^{-\lambda(t)\cdot t} \mathrm{d}t \tag{15-11}$$

当失效率为一个常数时，可得：

$$\theta = \frac{1}{\lambda} \tag{15-12}$$

当我们讨论完对软件可靠性的定量描述问题之后，需要对软件可靠度这个直接反映软件可靠性的度量指标作下列补充说明。

- 描述的软件对象必须明确，即需指明它与其他软件的界限。
- 软件失效必须明确定义。
- 必须假设硬件无故障（失效）和软件有关变量的输入值正确。
- 运行环境包括硬件环境、软件支持环境和确定的软件输入域。
- 规定的时间必须指明时间基准，可以是自然时间（日历时间）、运行时间、执行时间（CPU 时间），或其他时间基准。
- 软件无失效运行的机会通常以概率度量，但也可以模糊数学中的可能性加以度量。
- 上述定义是在时间域上进行的，这时软件可靠度是一种动态度量。也可以是在数据域上将软件可靠度定义为一种表态度量，表示软件成功执行一个回合的概率，软件回合（Run）是指软件在规定环境下的一个基本执行过程，如给定一组输入数据，到软件给定相应的输出数据这一过程。软件回合是软件运行的最小的、不可分的执行单位，软件的运行过程由一系列软件回合组成。
- 有时将软件运行环境简单地理解为软件运行剖面（operational profile）。欧空局（ESA）标准 PSS-01-21（1991）“ESA 软件产品保证要求”中，定义“软件运行剖面”为：“对系统使用条件的定义。系统的输入值都用其按时间的分布或按它们在可能输入范围内的出现概率的分布来定义”。简单来说，运行剖面定义了关于软件可靠性描述中的“规定条件”，也就相当于可靠性测试中需要考虑的测试环境、测试数据等一系列问题。

15.1.4　可靠性目标

前面我们定量分析软件的可靠性时，使用失效强度来表示软件缺陷对软件运行的影响程度。然而在实际情况中，对软件运行的影响程度不仅取决于软件失效发生的概率，还和软件失效的严重程度有很大关系。这里我们引出另外一个概念，失效严重程度类（failure severity class）。



失效严重程度类就是对用户具有相同程度影响的失效集合。

对失效严重程度的分级可以按照不同的标准进行，最为常见的是按对成本影响、对系统能力的影响等标准划分软件失效的严重程度类。

对成本的影响可能包括失效引起的额外运行成本、修复和恢复成本、现有或潜在的业务机会的损失等。由于失效严重程度类的影响分布很广泛，为了按照一定数量的等级去定义失效严重程度类，我们通常用数量级去划分等级。

表 15-1 给出了一个按照对成本的影响划分失效严重程度类的例子，这个例子涉及到的软件系统是某电子商务运营系统。

表 15-1 按照对成本的影响划分失效严重程度类

失效严重程度类	定义（人民币）/万元	失效严重程度类	定义（人民币）
1	＞100	4	1 千元～1 万元
2	10～100	5	＜1 千元
3	1～10		

对系统能力的影响常常表现为关键数据的损失、系统异常退出、系统崩溃、导致用户操作无效等。对于不同性质的软件系统，相同的表现可能造成的失效严重程度是不同的，比如对可用性要求较高的系统，导致长时间停机的失效常常会划分到较高的严重级别中去。

表 15-2 给出了一个按照对系统能力的影响划分失效严重程度类的例子，这个例子涉及到的软件系统是某电信实时计费系统。

表 15-2 按照对系统能力的影响划分失效严重程度类

失效严重程度类	定　义
1	系统崩溃，重要数据不能恢复
2	系统出错停止响应，重要数据可恢复
3	用户重要操作无响应，可恢复
4	部分操作无响应，但可用其他操作方式替代

有了失效严重程度的划分，我们现在可以结合失效强度来定量地表示一个软件系统的可靠性目标了。

可靠性目标是指客户对软件性能满意程度的期望。通常用可靠度、故障强度、平均无故障时间（MTTF）等指标来描述，根据不同项目的不同需要而定。建立定量的可靠性指标需要对可靠性、交付时间和成本进行平衡。为了定义系统的可靠性指标，必须确定系统的运行模式，定义故障的严重性等级，确定故障强度目标。

例如，对于表 15-2 的例子，我们可以根据经验和用户的需求确定软件系统需要达到的可靠程度，按照前面的公式，换算成失效强度和平均无失效时间，如表 15-3 所示。

表 15-3　可靠性目标参考表

失效严重程度类	可靠度要求/%	失效强度	平均无失效时间
1	99.9999	10^{-6}	114 年
2	99.99	10^{-4}	417 天
3	99	10^{-1}	4 天
4	90	1	9 个小时

15.1.5　可靠性测试的意义

软件可靠性问题已被越来越多的软件工程专家所重视，人们已开始投入大量的人力、物力去研究软件可靠性的设计、评估、测试等课题。以下多个方面可以反映出软件可靠性问题对软件工程实践，乃至对生产活动和社会活动产生的深远的影响。

- 软件失效可能造成灾难性的后果。一个最显著的例子就是由于控制系统 Fortran 程序中少了个逗点，致使控制系统未能发出正确的指令，最终使美国的一次宇宙飞行失败。而目前由于计算机和软件在各行各业中应用的日益广泛和深入，例如军用作战系统、民航指挥系统、银行支付系统、交通调控系统等，一旦发生严重级别的软件失效，轻则造成经济损失，重则危及人们的生命安全，危害国家安全。
- 软件的失效在整个计算机系统失效中的比例较高。某研究机构曾经作过统计，在计算机系统的失效中，有 80%和软件有关。原因是软件系统的内容结构太复杂了，一个较简单的程序，其所有的路径数就可能是一个天文数字。在软件开发的过程中，很难用全路径覆盖方式的测试去发现软件系统中隐藏的所有缺陷，也就是说，很难完全排除软件缺陷。
- 相比硬件可靠性技术，软件可靠性技术很不成熟，这就加剧了软件可靠性问题的重要性。例如在硬件可靠性领域，故障树分析（FTA）、失效模式与效应分析（FMEA）技术等比较成熟，容错技术也有广泛应用，但在软件可靠性领域，这些技术似乎尚未定型。
- 与硬件元器件成本急剧下降形成鲜明对比的是，软件费用呈有增无减的势头，而软件可靠性问题是造成这种费用增长的主要原因之一。
- 计算机技术获得日益广泛的应用，随着计算机应用系统中软件成分的不断增加，使得系统对于软件的依赖性越来越强，软件对生产活动和社会生活的影响



越来越大，从而增加了软件可靠性问题在软件工程领域乃至整个计算机工程领域的重要性。

软件可靠性问题的重要性也凸显出了，发展以发现软件可靠性缺陷为目的的可靠性测试技术的迫切性。

15.1.6 广义的可靠性测试与狭义的可靠性测试

广义的软件可靠性测试是指为了最终评价软件系统的可靠性而运用建模、统计、试验、分析、评价等一系列手段对软件系统实施的一种测试，一个完整的软件可靠性测试包括如图 15-1 所示的过程。

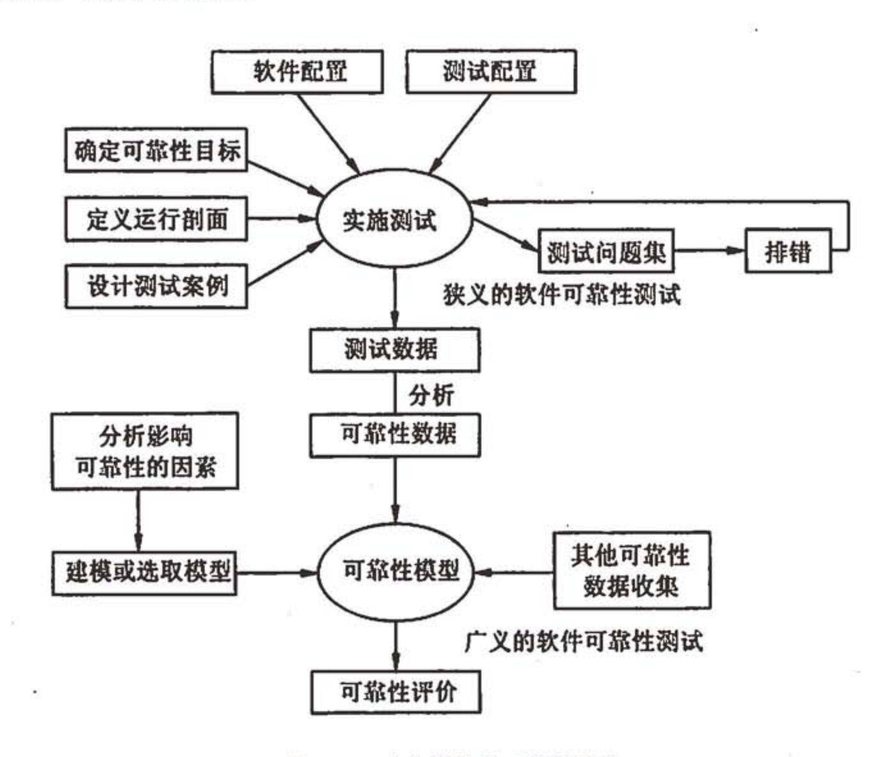


图 15-1 广义的软件可靠性测试

狭义的软件可靠性测试是指为了获取可靠性数据，按预先确定的测试用例，在软件的预期使用环境中，对软件实施的一种测试。狭义的软件可靠性测试也叫"软件可靠性试验（software reliability test）"，它是面向缺陷的测试，以用户将要使用的方式来测试软件，每一次测试代表用户将要完成的一组操作，使测试成为最终产品使用的预演。这就使得所获得的测试数据与软件的实际运行数据比较接近，可用于软件可靠性评价。

其实，软件可靠性测试是软件测试的一种形式，和易用性测试、性能测试、标准符合性测试等前面介绍的测试类型一样，是针对软件的某个重要质量特性，使用一定的测试用例对软件进行测试的过程。

可靠性测试是对软件产品的可靠性进行调查、分析和评价的一种手段。它不仅仅是为了用测试数据确定软件产品是否达到可靠性目标，还要对检测出的失效的分布、原因及后果进行分析，并给出纠正建议。总的来说，可靠性测试的目的可归纳为以下三个方面：

① 发现软件系统在需求、设计、编码、测试、实施等方面的各种缺陷。

② 为软件的使用和维护提供可靠性数据。

③ 确认软件是否达到可靠性的定量要求。

15.2　软件可靠性建模

15.2.1　影响软件可靠性的因素

在讲到软件可靠性评估的时候，我们不得不提到软件可靠性模型。软件可靠性模型（software reliability model）是指为预计或估算软件的可靠性所建立的可靠性框图和数学模型。建立可靠性模型是为了将复杂系统的可靠性逐级分解为简单系统的可靠性，以便于定量预计、分配、估算和评价复杂系统的可靠性。

为了构建软件的可靠性模型，我们首先要来分析一下影响软件可靠性的因素。影响软件可靠性的因素是纷杂而众多的，甚至包括技术以外的许多因素。首先我们必须只考虑影响软件可靠性的主要因素：缺陷的引入、发现和清除。缺陷的引入主要取决于软件产品的特性和软件的开发过程特性，软件产品的特性指软件本身的性质，开发过程特性包括开发技术、开发工具、开发人员的水平、需求的变化频度等。缺陷的发现依靠用户对软件的操作方式、运行环境等，也就是运行剖面。缺陷的清除依赖于失效的发现和修复活动及可靠性方面的投入。

从技术的角度来看，影响软件可靠性的主要因素如下。

- 运行剖面（环境）。软件可靠性的定义是相对运行环境而言的，一样的软件在不同的运行剖面下，其可靠性的表现是不一样的。
- 软件规模。也就是软件的大小，一个只有数十行代码的软件和几千万行代码的软件是不能相提并论的。
- 软件内部结构。结构对软件可靠性的影响主要取决于软件结构的复杂程度，一般来说，内部结构越复杂的软件，所包含的软件缺陷数就可能越多。



- 软件的开发方法和开发环境。软件工程表明，软件的开发方法对软件的可靠性有显著影响，例如，与非结构方法相比，结构化方法可以明显减少软件的缺陷数。
- 软件的可靠性投入。软件在生命周期中可靠性的投入包括开发者在可靠性设计、可靠性管理、可靠性测试、可靠性评价等方面投入的人力、资金、资源和时间等。经验表明，在早期重视软件可靠性并采取措施开发出来的软件，可靠性有明显的提高。

总之，有许许多多的因素影响着软件的可靠性，有些至今也无法确定它们与软件可靠性之间的定量关系，甚至定性关系也不甚清楚。

15.2.2 软件可靠性建模方法

一个软件可靠性模型通常（但不是绝对）由以下几部分组成。

- 模型假设。模型是实际情况的简化或规范化，总要包含若干假设，例如测试的选取代表实际运行剖面，不同软件失效独立发生等。
- 性能度量。软件可靠性模型的输出量就是性能度量，如失效强度、残留缺陷数等。在软件可靠性模型中性能度量通常以数学表达式给出。
- 参数估计方法。某些可靠性度量的实际值无法直接获得，例如残留缺陷数，这时需通过一定的方法估计参数的值，从而间接确定可靠性度量的值。当然，对于可直接获得实际值的可靠性度量，就无需参数估计了。
- 数据要求。一个软件可靠性模型要求一定的输入数据，即软件可靠性数据。不同类型的软件可靠性模型可能要求不同类型的软件可靠性数据。

绝大多数的模型包含三个共同假设。这些假设至今主宰着软件可靠性建模的研究发展，人们尚未找到克服这些假设局限性的有效方法。

- 代表性假设。此假设认为软件测试用例的选取代表软件实际的运行剖面，甚至认为测试用例是独立随机地选取。此假设实质上是指可以用测试产生的软件可靠性数据预测运行阶段的软件可靠性行为。
- 独立性假设。此假设认为软件失效是独立发生于不同时刻，一个软件失效的发生不影响另一个软件失效的发生。例如在概率范畴，假设相邻软件失效间隔构成一组独立随机变量，或假设一定时间内软件失效次数构成一个独立增量过程。在模糊数学范畴，则相邻软件失效间隔构成一组不相关的模糊变量。
- 相同性假设。此假设认为所有软件失效的后果（等级）相同，即建模过程只考虑软件失效的具体发生时刻，不区分软件的失效严重等级。

软件可靠性模型要描述失效过程对上一节所分析的因素的一般依赖形式。由于这些

因素大多数在本质上是概率性的，并且表现与时间相关联，所以我们通过失效数据的概率分布和随机过程随时间的变化的特性来整体区分软件可靠性模型。

我们常常通过下面估计或预测的方法来确定模型的参数。估计是通过收集到的失效数据进行统计分析，利用一定的推导过程归纳出模型的参数；预测则是使用软件产品自身的属性和开发过程来确定模型的参数，这种方法可以在开始执行程序前完成。

确定了模型的参数后，就可以来表示失效过程的很多不同的特性。例如大多数模型都会对如下的内容进行解析表达。

- 任何时间点所经历的平均失效数。
- 一段时间间隔内的平均失效数。
- 任何时间点的失效强度。
- 失效区间的概率分布。

在对将来的故障行为进行预测时，应保证模型参数的值不发生变化。如果在进行预测时发现引入了新的错误，或修复行为使新的故障不断发生，就应停止预测，并等足够多的故障出现后，再重新进行模型参数的估计。否则，这样的变化会因为增加问题的复杂程度而使模型的实用性降低。

一般来说，软件可靠性模型是以在固定不变的运行环境中运行的不变的程序作为估测实体的。这也就是说，程序的代码和运行剖面都不发生变化，但它们往往总要发生变化的，于是在这种情况之下，就应采取分段处理的方式来进行工作。因此，模型主要集中注意力于排错。但是，也有的模型具有能处理缓慢地引进错误情况的能力。

对于一个已发行并正在运行的程序，应暂缓安装新的功能和对下一次发行的版本的修复。如果能保持一个不变的运行剖面，则程序的故障密度将显示为一个常数。

一般来说，一个好的软件可靠性模型增加了关于开发项目的交流，并对了解软件开发过程提供了一个共同的工作基础。它也增加了管理的透明度和其他令人感兴趣的东西。即使在特殊的情况之下，通过模型作出的预测并不是很精确的话，上面的这些优点也仍然是明显而有价值的。

要建立一个有用的软件可靠性模型必须有坚实的理论研究工作、有关工具的建造和实际工作经验的积累。通常这些工作要许多人一年的工作量。相反，要应用一个好的软件可靠性模型，则要求以极少的项目资源就可以在实际工作中产生好的效益。

一个好的软件可靠性模型应该具有如下重要特性。

- 基于可靠的假设；
- 简单；
- 计算一些有用的量；
- 给出未来失效行为的好的映射；

- 可广泛应用。

15.2.3 软件的可靠性模型分类

一个有效的软件可靠性模型应尽可能地将上面所述的因素在软件可靠性建模时加以考虑，尽可能简明地反映出来。自 1972 年第一个软件可靠性分析模型发表的 20 多年来，见之于文献的软件可靠性统计分析模型将近百种。这些可靠性模型大致可分为如下 10 类。

- 种子法模型；
- 失效率类模型；
- 曲线拟合类模型；
- 可靠性增长模型；
- 程序结构分析模型；
- 输入域分类模型；
- 执行路径分析方法模型；
- 非齐次泊松过程模型；
- 马尔可夫过程模型；
- 贝叶斯类模型。

下面分别对这些模型进行简单介绍。

（1）种子法模型

这类模型利用捕获–再捕获抽样技术估计程序中的错误数，在程序中预先有意“播种”一些设定的错误“种子”，然后根据测试出的原始错误数和发现的诱导错误的比例，来估计程序中残留的错误数。其优点是简便易行，缺点是诱导错误的“种子”与实际的原始错误之间的类比性估量困难。

（2）失效率类模型

这类模型用来研究程序的失效率的，主要有下列内容。

- Jelinski-Moranda 的 de-eutrophication 模型；
- Jelinski-Moranda 的几何 de-eutrophication 模型；
- Schick-Wolverton 模型；
- 改进的 Schick-Wolverton 模型；
- Moranda 的几何泊松模型；
- Goal 和 Okumoto 不完全排错模型。

（3）曲线拟合类模型

这类模型用回归分析的方法研究软件复杂性、程序中的缺陷数、失效率、失效间隔

时间，包括参数方法和非参数方法两种。

（4）可靠性增长模型

这类模型预测软件在检错过程中的可靠性改进，用增长函数来描述软件的改进过程。这类模型如下。

- Duane 模型；
- Weibull 模型；
- Wagoner 的 Weibull 改进模型；
- Yamada 和 Osaki 的逻辑增长曲线；
- Gompertz 的增长曲线。

（5）程序结构分析模型

程序结构模型是根据程序、子程序及其相互间的调用关系，形成一个可靠性分析网络。网络中的每一结点代表一个子程序或一个模块，网络中的每一有向弧代表模块间的程序执行顺序。假定各结点的可靠性是相互独立的，通过对每一个结点可靠性、结点间转换的可靠性和网络在结点间的转换概率，得出该持续程序的整体可靠性。这类模型如下。

- Littewood 马尔可夫结构模型；
- Cheung 的面向用户的马尔可夫模型。

（6）输入域分类模型

这类模型选取软件输入域中的某些样本“点”运行程序，根据这些样本点在“实际”使用环境中的使用概率的测试运行时的成功/失效率，推断软件的使用可靠性。这类模型的重点（亦是难点）是输入域的概率分布的确定及对软件运行剖面的正确描述。这类模型如下。

- Nelson 模型；
- Bastani 的基于输入域的随机过程模型。

（7）执行路径分析方法模型

这类模型的分析方法与上面的模型相似，先计算程序各逻辑路径的执行概率和程序中错误路径的执行概率，再综合出该软件的使用可靠性。Shooman 分解模型属于此类。

（8）非齐次泊松过程模型

非齐次泊松过程模型，即 NHPP，是以软件测试过程中单位时间的失效次数为独立泊松随机变量，来预测在今后软件的某使用时间点的累计失效数。这类模型如下。

- Musa 的指数模型；
- Goel 和 Okumoto 的 NHPP 模型；
- S-型可靠性增长模型；
- 超指数增长模型；



- Pham 改进的 NHPP 模型。

（9）马尔可夫过程模型

这类模型如下。

- 完全改错的线性死亡模型；
- 不完全改错的线性死亡模型；
- 完全改错的非静态线性死亡模型。

（10）贝叶斯类模型

这是利用失效率的试验前分布和当前的测试失效信息，来评估软件的可靠性。这是一类当软件可靠性工程师对软件的开发过程有充分的了解，软件的继承性比较好时具有良好效果的可靠性分析模型。这类模型如下。

- 连续时间的离散型马尔可夫链；
- Shock 模型。

另外，Musa 和 Okumoto 依据模型的不同属性对可靠性模型进行以下分类。

- 时间域：有两种，自然或日历时间与执行（CPU）时间；
- 失效数类：取决于无限时间内发生的失效数是有限的还是无限的；
- 失效数分布：相对于时间系统失效数的统计分布形式，主要的两类是泊松分布型和二项分布型；
- 有限类：对有限失效数的类别适用，用时间表示的失效强度的函数形式；
- 无限类：对无限失效数的类别适用，用经验期望失效数表示的失效强度的函数形式。

15.2.4 软件可靠性模型举例

迄今已有数十种模型是根据上一小节中关于模型的分类方法进行的分类，下面我们将介绍 Jelinski-Moranda 模型的基本思想及其相关的历史背景。

Jelinski-Moranda 模型经常简称为 JM 模型，它是 Z.Jelinski 和 P.Moranda 于 1972 年提出的软件可靠性数学模型，是最具代表性的早期软件可靠性马尔可夫过程的数学模型。随后的许多工作，都是在它的基础上，对其中与软件开发实际不相适合的地方进行改进而提出来的，所以，JM 模型是具有广泛影响的模型之一。

1．模型假设

JM 模型的基本假设如下。

- 软件系统中的初始错误个数为一个未知的常数，用 N_0 表示。
- 可靠性测试中发现的错误立即被完全排除，并且排除过程不引入新的错误，排除时间忽略不计。因此，每次排错之后，N_0 就要减去 1。

- 在任何一个失效间隔区间，软件系统的失效率与系统中剩余的错误个数成正比，比例常数用Φ表示。

其实，最初 Jelinski 和 Moranda 提出的模型假设只有最后一条，前面两个假设是后人根据使用过程中出现的问题归纳总结而来的。

2．函数表达式

根据假设，每发生 1 次失效，错误数都要减去 1，如果用 t_1，t_2，…，t_i 表示从 0 时刻开始的每次失效间隔时间，那第 i–1 次失效到第 i 次失效之间的失效率就是：

$$\lambda(t_i)=\Phi(N_0-i+1) \tag{15-13}$$

根据我们在可靠性定量描述一节的讨论，我们知道失效强度函数为：

$$f(t_i)=\Phi(N_0-i+1)e^{-\Phi(N_0-i+1)t_i} \tag{15-14}$$

可靠度函数为：

$$R(t_i)=e^{-\Phi(N_0-i+1)t_i} \tag{15-15}$$

失效概率分布函数为：

$$F(t_i)=1-e^{-\Phi(N_0-i+1)t_i} \tag{15-16}$$

3．参数估计

在可靠度函数表达式中含有两个未知参数Φ和 N_0，下面我们运用统计学中的最大似然法来对参数Φ和 N_0进行估算。

由公式（15-15）可得似然函数：

$$L(t_1,t_2,\cdots,t_n)=\prod_{i=1}^{n}\Phi(N_0-i+1)e^{-\Phi(N_0-i+1)t_i} \tag{15-17}$$

对公式（15-17）取对数，得到对数似然函数：

$$\begin{aligned}\ln L&=\sum_{i=1}^{n}\ln\left[\Phi(N_0-i+1)e^{-\Phi(N_0-i+1)t_i}\right]\\&=\sum_{i=1}^{n}\ln[\Phi(N_0-i+1)]-\sum_{i=1}^{n}[\Phi(N_0-i+1)t_i]\\&=\sum_{i=1}^{n}\ln(N_0-i+1)+n\ln\Phi-\sum_{i=1}^{n}[\Phi(N_0-i+1)t_i]\end{aligned} \tag{15-18}$$

对公式（15-18）分别对 N_0和Φ求偏导，并令结果为零：

$$\frac{\partial\ln L}{\partial N_0}=\sum_{i=1}^{n}\frac{1}{N_0-i+1}-\sum_{i=1}^{n}\Phi t_i=0 \tag{15-19}$$

$$\frac{\partial\ln L}{\partial\Phi}=\frac{n}{\Phi}-\sum_{i=1}^{n}(N_0-i+1)t_i=0 \tag{15-20}$$

公式 15-19 可以写成：

$$\frac{\partial \ln L}{\partial N_0}=\sum_{i=1}^{n}\frac{1}{N_0-i+1}$$
$$=\frac{nT}{N_0T-\sum_{i=1}^{n}(i-1)t_i} \tag{15-21}$$
$$=\frac{n}{N_0-\frac{1}{T}\sum_{i=1}^{n}(i-1)t_i}$$

公式 15-21 中不含Φ，因而可以由测试收集的数据，计算出 $T=\sum_{i=1}^{n}t_i$ 和 $\sum_{i=1}^{n}(i-1)t_i$ 的值，将它们代入公式 15-21 中，即可先解出 N_0 的估计值 $\hat{N}_0$：

$$\hat{N}_0=\frac{1}{N_0}+\frac{1}{N_0-1}+\cdots+\frac{1}{N_0-(n-1)}$$
$$=\frac{n}{N_0-\sum_{i=1}^{n}t_i\cdot\sum_{i=1}^{n}(i-1)t_i} \tag{15-22}$$

我们再来解出另一个参数Φ的估计值，令

$$T=\sum_{i=1}^{n}t_i \tag{15-23}$$

则从 15-20 中可解出

$$\Phi=\frac{n}{N_0T-\sum_{i=1}^{n}(i-1)t_i} \tag{15-24}$$

代入 N_0 的估计值 $\hat{N}_0$，可解出Φ的估计值$\hat{\Phi}$：

$$\hat{\Phi}=\frac{n}{\hat{N}_0\sum_{i=1}^{n}t_i-\sum_{i=1}^{n}(i-1)t_i} \tag{15-25}$$

需要说明的是，软件可靠性是一门正在发展中的分支学科，许多来源于硬件可靠性的理论在软件可靠性研究中并不适用，有关软件可靠性的模型并不成熟，并且应用范围也非常有限，软件可靠性的定量分析方法和数学模型要在实践中不断加以验证和修正，对于不同类型的软件，模型的假设、表示公式及应用方式也有很大的区别。

15.3　软件可靠性测试

15.3.1　软件的可靠性测试概述

软件测试者可以使用很多方法进行软件测试，如按行为或结构来划分输入域的划分测试，纯粹随机选择输入的随机测试，基于功能、路径、数据流或控制流的覆盖测试等。对于给定的软件，每种测试方法都局限于暴露一定数量和一些类别的缺陷。通过这些测试能够查找、定位、改正和消除某些缺陷，实现一定意义上的软件可靠性增长。但是，由于它们都是面向错误的测试，测试所得的结果数据不能直接用于软件可靠性评价，必须经过一定的分析处理后方可使用可靠性模型进行可靠性评价。

软件可靠性测试由可靠性目标的确定、运行剖面的开发、测试用例的设计、测试实施、测试结果的分析等主要活动组成。

软件可靠性测试还必须考虑对软件开发进度和成本的影响，最好是在受控的自动测试环境下，由专业测试机构完成。

软件可靠性测试是一种有效的软件测试和软件可靠性评价技术。尽管软件可靠性测试也不能保证软件中残存的缺陷数最少，但经过软件可靠性测试可以保证软件的可靠性达到较高的要求，对于开发高可靠性与高安全性软件系统很有帮助。

软件可靠性测试要在工程上获得广泛应用，还有许多实际问题需要解决。

15.3.2　定义软件运行剖面

定义运行剖面首先需要为软件的使用行为建模，建模可以采用马尔可夫链来完成。用马尔可夫链将输入域编码为一个代表用户观点的软件使用的状态集。弧用来连接状态并表示由各种激励导致的转换。这些激励可能由硬件、人机接口或其他软件等产生。将转换概率分配给每个弧，用来代表一个典型用户最有可能施加给系统的激励。这种类型的马尔可夫链是一个离散的有限状态集。这类模型可以用有向图或转换矩阵表示。

定义运行剖面的下一步是开发使用模型，明确需要测试的内容。软件系统可能会有许多用户和用户类别，每类用户都可能以不同的方式使用系统。开发使用模型涉及到将输入域分层。有两种类型的分层形式：用户级分层和用法级分层。 用户级分层依赖于谁或什么能激励系统。用法级分层依赖于在测试状态下系统能做什么。换句话说，用户级分层考虑各种类型的用户以及他们如何使用系统，用法级分层则要求考虑系统能够提供的所有功能。一旦用户和用法模型被开发出来，弧上的概率将被分配。这些概率估计主要是基于如下几个方面。



- 从现有系统收集到的数据；
- 与用户的交谈或对用户进行观察获得的信息；
- 原型使用与测试分析的结果；
- 相关领域专家的意见。

定义使用概率的最佳方法是使用实际的用户数据，如来自系统原型、前一版本的使用数据；其次是由该软件应用领域的用户和专家提供的预期使用数据；在没有任何数据可用的情况下，只能是将每个状态现有的弧分配相同的概率，这是最差的一种方法。

由于软件可靠性行为是相对于软件实际的运行剖面而言的，同一软件在不同运行剖面下其可靠性表现可能大不相同，所以用于可靠性测试准备的运行剖面的开发与定义必须充分分析和考虑软件的实际运行情况。

软件可靠性测试假设每个操作的数据输入都有同样的发生错误的概率，这样最频繁出现的操作和输入将表现出最高的故障率。对于特定的操作环境这是正确的，但无法贯穿系统的全部操作集合。典型的例子是飞机的飞行控制软件，在正常飞行、起飞、降落、地面运动和地面等待这五个状态中，尽管起飞和降落在运行剖面上只占有很小的百分比，但是它们却占有很大的故障比例。对于高安全性要求的软件，一个看起来很少使用的代码路径也可能带来灾难性的后果。因此，对于边界、跃迁情况和关键功能不应该用简单的运行剖面来对待，应该构造专门的运行剖面，补充统计模型之外的测试用例。在覆盖率水平不够时，可根据具体空白，进行适当的补充测试，如果补充测试发现了错误，就可分析这些错误，估计其对可靠性产生的影响。

一个产品有可能需要开发多个运行剖面，这取决于它所包含的运行模式和关键操作，通常需要为关键操作单独定义运行剖面。

15.3.3 可靠性测试用例设计

为了对软件可靠性进行良好的预计，必须在软件的运行域上对其进行测试，首先定义一个相应的剖面来镜像运行域，然后使用这个剖面驱动测试，这样可以使测试真实地反映软件地使用情况。

由于可能的输入几乎是无限的，测试必须从中选择出一些样本，即测试用例。测试用例要能够反映实际的使用情况，反映系统的运行剖面。将统计方法运用到运行剖面开发和测试用例生成中去，并为在运行剖面中的每个元素都定量地赋予一个发生概率值和关键因子，然后根据这些因素分配测试资源，挑选和生成测试用例。

在这种测试中，优先测试那些最重要或最频繁使用的功能，释放和缓解最高级别的风险，有助于尽早发现那些对可靠性有最大影响的故障，以保证软件的按期交付。

设计测试用例就是针对特定功能或组合功能设计测试方案，并编写成文档。测试用

例的选择既要有一般情况，也应有极限情况以及最大和最小的边界值情况。因为测试的目的是暴露应用软件中隐藏的缺陷，所以在设计选取测试用例和数据时要考虑那些易于发现缺陷的测试用例和数据，结合复杂的运行环境，在所有可能的输入条件和输出条件中确定测试数据，来检查应用软件是否都能产生正确的输出。

一个典型的测试用例应该包括下列组成部分。

- 测试用例标识；
- 被测对象；
- 测试环境及条件；
- 测试输入；
- 操作步骤；
- 预期输出；
- 判断输出结果是否符合的标准；
- 测试对象的特殊需求。

由于可靠性测试的主要目的是评估软件系统的可靠性，因此，除了常规的测试用例集仍然适用外，我们还要着重考虑和可靠性密切相关的一些特殊情况。在测试中，可以考虑进行“强化输入”，即比正常输入更恶劣（合理程度的恶劣）的输入。

表 15-4 给出一些参考例子：

表 15-4　可靠性测试用例设计时重点考虑的一些特殊情况

序　号	测 试 项 目	描　　述
1	屏蔽用户操作错误	考察对用户常见的误操作的提示和屏蔽情况
2	错误提示的准确性	对用户的错误提示准确程度
3	错误是否导致系统异常退出	有无操作错误引起系统异常退出的情况
4	数据可靠性	系统应对输入的数据进行有效性检查，对冗余的数据进行过滤、校验和清洗，保证数据的正确性和可靠性
5	异常情况的影响	考察数据和系统的受影响程度，若受损，是否提供补救工具，补救的情况如何，异常情况包括： ① 硬件故障； ② 网络故障； ③ 部分软件模块失效。

15.3.4　可靠性测试的实施

在进行应用软件的可靠性测试前有必要检查软件需求与设计文档是否一致，检查软件开发过程中形成的文档的准确性、完整性以及与程序的一致性，检查所交付程序和数

据以及相应的软件支持环境是否符合要求。

这些检查虽然增加了工作量，但对于在测试早期发现错误和提高软件的质量是非常必要的。

软件可靠性测试必须是受控测试，在运行此类测试时，为了保证统计数据的有效性，测试过程中的每个测试用例必须施加于相同的软件版本，新的软件版本意味着新测试的开始。

在有些情况下，不能进行纯粹的可靠性测试。因为客户的要求、合同的规定或者标准的约束，需要补充其他形式的非统计测试。这时的最佳选择是既执行可靠性测试，也执行非统计测试（如覆盖测试）。如果非统计测试在可靠性测试之前完成，由统计测试产生的统计数据仍然有效。但是在可靠性测试之后执行非统计测试，可能会影响软件可靠性评估的准确性。

软件可靠性测试同样依赖于软件的可测试性。可靠性测试难点就在于判断测试用例的运行是成功还是失败。在控制系统及类似的软件中，失效由详细说明、时间（通常是CPU时间或时钟时间）来客观地定义。而在一般应用系统中，失效的定义更主观些，它不仅依赖于程序是否符合规格说明的要求，也取决于指定的性能是否能够达到用户的期望，但是否达到期望没有确定的标准。在一些科学计算中，计算结果只能由计算机给出，在这种情况下，如果软件只是输出了错误的结果而不是整个系统发生失效，错误就不可能被发现。此时可以将测试分成两个阶段进行。第一阶段运行较少量的测试用例，并对照规范进行仔细检查。第二阶段再运行大量测试用例。第二阶段不用人工检查输出的每项内容，而是找失效现象，包括错误信息、断电、崩溃和死机。也可把输出记录到文件中，采用搜索或过滤方法进行处理。如果软件有足够的可测试性，这种方法不会漏掉很多的严重失效。如果计算的正确性无法验证，就需要对软件进行一些形式化的证明。

开发方交付的任何软件文档中与可靠性质量特性有关的部分、程序以及数据都应当按照需求说明和质量需求进行测试。在项目合同、需求说明书和用户文档中规定的所有配置情况下，程序和数据都必须进行测试。

软件可靠性数据是可靠性评价的基础。为了获得更多的可靠性数据，应该使用多台计算机同时运行软件，以增加累计运行时间。应该建立软件错误报告、分析与纠正措施系统。按照相关标准的要求，制定和实施软件错误报告和可靠性数据收集、保存、分析和处理的规程，完整、准确地记录软件测试阶段的软件错误报告和收集可靠性数据。

用时间定义的软件可靠性数据可以分为4类，具体如下。

- 失效时间数据，记录发生一次失效所累积经历的时间；
- 失效间隔时间数据，记录本次失效与上一次失效间的间隔时间；
- 分组时间内的失效数，记录某个时间区内发生了多少次失效；

- 分组时间的累积失效数，记录到某个区间的累积失效数。

这 4 类数据可以互相转化。

在测试过程中必须真实地进行记录，每个测试记录必须包含如下信息。

- 测试时间；
- 含有测试用例的测试说明或标识；
- 所有与测试有关的测试结果，包括失效数据；
- 测试人员。

测试活动结束后要编写《软件可靠性测试报告》，对测试用例及测试结果在测试报告中加以总结归纳。编写时可以参考 GJB 438A-97 中提供的《软件测试报告》格式，并应根据情况进行剪裁。测试报告应具备如下内容。

- 软件产品标识；
- 测试环境配置（硬件和软件）；
- 测试依据；
- 测试结果；
- 测试问题；
- 测试时间。

把可靠性测试过程进行规范化，有利于获得真实有效的数据，为最终得到客观的可靠性评价结果奠定基础。

对于软件可靠性测试用例的设计，有关更详细的内容可以参照本书中“黑盒测试用例设计”部分。

15.4　软件可靠性评价

15.4.1　软件可靠性评价概述

软件可靠性评价是软件可靠性活动的重要组成部分，既适用于软件开发过程，也可针对最终软件系统。在软件开发过程中使用软件可靠性评价，可以使用软件可靠性模型，估计软件当前的可靠性，以确认是否可以终止测试并发布软件，同时还可以预计软件要达到相应的可靠性水平所需要的时间和工作量，评价提交软件时的软件可靠性水平。对于最终软件产品，软件可靠性评价结合可靠性验证测试，确认软件的执行与需求的一致性，确定最终软件产品所达到的可靠性水平。

这一节阐述的软件可靠性评价工作是指选用或建立合适的可靠性数学模型，运用统计技术和其他手段，对软件可靠性测试和系统运行期间收集的软件失效数据进行处理，

并评估和预测软件可靠性的过程。这个过程包含 3 个方面：

① 选择可靠性模型；

② 收集可靠性数据；

③ 可靠性评估和预测。

15.4.2 怎样选择可靠性模型

在前面我们讨论了软件的可靠性模型以及一个举例，一些可靠性研究者试图寻找一个最好的模型，能适用于所有的软件系统，但这样的工作是徒劳的。因为对于不同的软件系统，出于不同的可靠性分析目的，模型的适用性是不一样的。但究竟怎样来为可靠性评价选用不同的模型，却又是一个不小的难题。

针对可靠性模型的构成以及我们使用模型来进行可靠性评价的目的，可以从以下几个方面进行比较和选择。

1．模型假设的适用性

模型假设是可靠性模型的基础，模型假设要符合软件系统的现有状况，或与假设冲突的因素在软件系统中应该是可忽略的。例如，有的模型假定检测或发现的软件缺陷是立即排除掉的，而且排除时间忽略不计，如果现有的软件系统对于严重程度类较低的软件缺陷不进行立即排错，那么这个模型显然是不适用的。

往往一个模型的假设有许多条，我们需要在选用模型的时候对每一条假设进行细致的分析，评估现有的软件系统中不符合假设的因素对可靠性评价的影响如何，以确定模型是否适合软件系统的可靠性评价工作。

2．预测的能力与质量

预测的能力与质量是指模型根据现在和历史的可靠性数据，预测将来的可靠性和失效概率的能力，以及预测结果的准确程度。显然模型预测的能力与质量是比较难于评价的，但任何一个模型只有在实践中加以实验和不断改善，才能得到认可。所以，我们在满足其他条件的前提下，应尽量选用比较成熟、应用较广的模型作为分析模型。

3．模型输出值能否满足可靠性评价需求

我们使用模型进行可靠性评价的最终目的，是想得到软件系统当前的可靠性定量数据，和预测一定时间后的可靠性数据，我们可以根据可靠性测试目的来确定哪些模型的输出值满足可靠性评价需求。一般说来，最重要的几个需要精确估计的可靠性定量指标包括如下内容。

- 当前的可靠度；
- 平均无失效时间（MTTF）；
- 故障密度；

- 期望达到规定可靠性目标的日期；
- 达到规定的可靠性目标的成本要求。

4．模型使用的简便性

模型使用的简便性一般包含三层含义。

- 模型需要的数据在软件系统中应该易于收集，而且收集需要投入的成本不能超过可靠性计划的预算。
- 模型应该简单易懂，进行可靠性分析的软件测试人员不会花费太多的时间去研究专业的数学理论，他们只需要知道哪些假设适用，需要收集哪些数据，能够得到哪些分析结果就可以了。
- 模型应该便于使用，最好能用工具实现数据的输入。也就是说，测试人员除了输入可靠性数据外，不需要深入模型内部进行一些额外的工作。

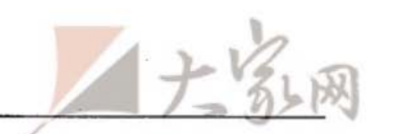
尽管这样，由于可靠性研究理论在软件工程领域发展的限制，可供选择的可靠性模型极其有限，这已在相当大的程度上制约了可靠性测试的开展。

15.4.3　可靠性数据的收集

面向缺陷的可靠性测试产生的测试数据经过分析后，可以得到非常有价值的可靠性数据，是可靠性评价所用数据的一个重要来源，这部分数据取决于定义的运行剖面和选取的测试用例集。可靠性数据主要是指软件失效数据，是软件可靠性评价的基础，主要是在软件测试、实施阶段收集的，在软件工程的需求、设计和开发阶段的可靠性活动，也会产生影响较大的其他可靠性数据，因此，可靠性数据的收集工作是贯穿于整个软件生命周期的。

由于软件开发过程中的特殊复杂性及许多潜在因素的影响，可靠性数据收集工作会极为困难，目前，关于数据的收集工作，存在许多有待解决的问题。

- 可靠性数据的规范不统一。对软件进行度量的定义混乱不清。例如，时间、缺陷、失效、模型结构等的定义，就相当含糊，缺乏统一的标准。这样就使得在进行软件可靠性数据的收集时，目标不明确，甚至无从下手。
- 数据收集工作的连续性不能保证。可靠性数据的收集是连续的长期的过程，而且需要投入一定的资金、人力、时间，往往这些投入会在软件的开发计划中被忽略，以至于不能保证可靠性数据收集工作的正常进行。
- 缺乏有效的数据收集手段。进行数据收集同样需要方便实用的工具，然而除了在可靠性测试方面有了一些可用的数据收集工具外，其他方面的工具还十分缺乏。
- 数据的完整性不能保证。即使可靠性活动计划作得再周密，收集到的数据仍有



可能是不完全的，而且遗漏的数据往往会影响到可靠性评价的结果。

- 数据质量和准确性不能保证。不完全的排错及诊断，使收集的数据中含有不少虚假的成分，它们不能正确反映软件的真实状况。使用不准确的可靠性数据进行的可靠性评价，误差有可能会比利用可靠性模型进行预测产生的误差大一个数量级，这说明数据质量的重要性。

为了给软件可靠性评价提供一套准确、有效的可靠性数据，有必要在软件工程中重视软件可靠性数据的收集工作，采取一些措施尽量解决上述问题。在现有条件下，可行的一些办法如下。

- 及早确定所采用的可靠性模型，以确定需要收集的可靠性数据，并明确定义可靠性数据规范中的一些术语和记录方法，如时间、失效、失效严重程度类的定义，制定标准的可靠性数据记录和统计表格等。
- 制定可实施性较强的可靠性数据收集计划，指定专人负责，抽取部分开发人员、质量保证人员、测试人员、用户业务人员参加，按照统一的规范收集记录可靠性数据。
- 重视软件测试特别是可靠性测试产生的测试数据的整理和分析，因为这部分数据是用模拟软件实际运行环境的方法，模拟用户实际操作的测试用例测试软件系统产生的数据，对软件可靠性评价和预测有较高的实用价值。
- 充分利用数据库来完成可靠性数据的存储和统计分析。一方面减少数据管理的混乱，一方面提高数据处理的效率。

15.4.4 软件可靠性的评估和预测

软件可靠性的评估和预测的主要目的，是为了评估软件系统的可靠性状况和预测将来一段时间的可靠性水平。下面是一些常见的需要利用软件可靠性评价进行解答的问题。

- 判断是否达到了可靠性目标，是否达到了软件付诸使用的条件，是否达到了中止测试的条件。
- 如未能达到，要再投入多少时间、多少人力和多少资金，达到可靠性目标或投入使用。
- 在软件系统投入实际运行一年或若干时间后，经过维护、升级、修改，软件能否达到交付或部分交付用户使用的可靠性水平。

目前有不少支持软件可靠性估计的软件工具，我们只要将收集的失效数据分类并录入，选择合适的可靠性模型就可以获得软件可靠性的评价结果。

然而，对于那些可靠性要求很高的系统，必须进行很多测试才能预计出高置信度的可靠性结果，即便如此，仍然可能没有任何失效发生。没有失效就无法估计可靠性，不

能认为程序的可靠性是 1.0。除非我们已经进行了完全的测试，否则程序不失效我们就无法做出估计，而完全的测试几乎总是不可能的。如果在测试期间没有失效发生，我们可以简单地假设测试是基于二项式分布的，这样就可以对可靠性作保守估计，也可以凭经验，根据无故障运行的测试用例的数量，在一定的置信度水平上，估计可靠性的等级。

软件可靠性评价技术和方法主要依据选用的软件可靠性模型，其来源于统计理论。软件可靠性评估和预测以软件可靠性模型分析为主，但也要在模型之外运用一些统计技术和手段对可靠性数据进行分析，作为可靠性模型的补充、完善和修正。这些辅助方法如下。

- 失效数据的图形分析法。

运行图形处理软件失效数据，可以直观地帮助我们进行分析。图形指标如下。

① 累积失效个数图形；

② 单位时间段内的失效数的图形；

③ 失效间隔时间图形。

- 试探性数据分析技术（EDA）。

对于失效数据图形进行一定的数字化分析，能发现和揭示出数据中的异常。对可靠性分析有用的如下。

① 循环相关；

② 短期内失效数的急剧上升；

③ 失效数集中的时间段。

这种分析方法常可以发现因排错引入新的缺陷、数据收集的质量问题及时间域的错误定义等问题。

还有其他一些分析方法，这里就不一一赘述了。

15.5 软件的可靠性设计与管理

15.5.1 软件可靠性设计

在测试阶段我们利用测试手段收集测试数据，并利用软件可靠性模型，可以评估或预测软件的可靠性，这些软件可靠性测试活动虽然能通过查错-排错活动有限地改善软件可靠性，但不能从根本上提高软件的可靠性，也难以保证软件可靠性，并且修改由于设计导致的软件缺陷，有可能付出比较昂贵的代价。实践证明，保障软件可靠性最有效、最经济、最重要的手段是在软件设计阶段采取措施进行可靠性控制。为了从根本上提高软件的可靠性，降低软件后期修改的成本和难度，人们提出了可靠性设计的概念。



可靠性设计其实就是在常规的软件设计中，应用各种方法和技术，使程序设计在兼顾用户的功能和性能需求的同时，全面满足软件的可靠性要求，即采用一些技术手段，把可靠性“设计”到软件中去。软件可靠性设计技术就是以提高和保障软件的可靠性为目的，在软件设计阶段运用的一种特殊的设计技术。

在软件工程中已有很多比较成熟的设计技术，如结构化设计、模块化设计、自顶向下设计、自底向上设计等，这些技术是为了保障软件的整体质量而采用的，在此基础上，为了进一步提高软件的可靠性，通常会采用一些特殊设计技术。虽然软件可靠性设计技术与普通的软件设计技术没有明显的界限，但软件可靠性设计仍要遵循一些自己的原则：

① 软件可靠性设计是软件设计的一部分，必须在软件的总体设计框架中使用，并且不能与其他设计原则相冲突。

② 软件可靠性设计在满足提高软件质量要求的前提下，以提高和保障软件可靠性为最终目标。

③ 软件可靠性设计应确定软件的可靠性目标，不能无限扩大化，并且排在功能度、用户需求、开发费用之后考虑。

可靠性设计概念被广为引用，但并没有多少人能提出非常实用并且广泛运用的可靠性设计技术。一般来说，被认可的且具有应用前景的软件可靠性设计技术主要有容错设计、检错设计、降低复杂度设计等技术。

1. 容错设计技术

对于软件失效后果特别严重的场合，如飞机的飞行控制系统、空中交通管制系统、核反应堆安全控制系统等，可采用容错设计方法。常用的软件容错技术主要有三种方法：恢复块设计、N 版本程序设计和冗余设计。

- 恢复块设计。

程序的执行过程可以看成是由一系列操作构成的，这些操作又可由更小的操作构成。恢复块设计就是选择一组操作作为容错设计单元，从而把普通的程序块变成恢复块。被选择用来构造恢复块的程序块可以是模块、过程、子程序、程序段等。

一个恢复块包含有若干个功能相同、设计差异的程序块文本，每一时刻有一个文本处于运行状态。一旦该文本出现故障，则用备份文本加以替换，从而构成“动态冗余”。软件容错的恢复块方法就是使软件包含有一系列恢复块。

- N 版本程序设计。

N 版本程序的核心是通过设计出多个模块或不同版本，对于相同初始条件和相同输入的操作结果，实行多数表决，防止其中某一软件模块/版本的故障提供错误的服务，以实现软件容错。为使此种容错技术具有良好的结果，必须注意以下两个方面：

① 使软件的需求说明具有完全性和精确性。这是保证软件设计错误不相关的前提。

因为软件的需求说明是不同设计组织和人员的惟一共同出发点。

② 设计全过程的不相关性。它要求各个不同的软件设计人员彼此不交流，程序设计使用不同的算法、不同的编程语言、不同的编译程序、不同的设计工具、不同的实现方法和不同的测试方法。为了彻底保证软件设计的不相关性，甚至提出设计人员应具有不同的受教育背景，来自不同地域、不同的国家。

- 冗余设计。

改善软件可靠性的一个重要技术是冗余设计。在硬件系统中，在主运行的系统之外备用额外的元件或系统，如果出现一个元件故障或系统故障，则立即更换冗余的元件或切换到冗余的系统，则该硬件系统仍可以维持运行。在软件系统中，冗余技术的运用有所区别。如果我们采用相同两套软件系统作为互为备份，其意义不大，因为在相同的运行环境中，一套软件出故障的地方，另外一套也一定会出现故障。软件的冗余设计技术实现的原理是在一套完整的软件系统之外，设计一种不同路径、不同算法或不同实现方法的模块或系统作为备份，在出现故障时可以使用冗余的部分进行替换，从而维持软件系统的正常运行。

从表面上看，设计开发完成同样功能但实现方法完全不同的两套软件系统，需要的费用可能接近于单个版本软件开发费用的两倍，但采用冗余技术设计软件所增加的额外费用肯定远低于重新设计一个版本软件的费用。这是因为大多数设计花费，例如文档、测试以及人力都是有可能复用的。冗余设计还有可能导致软件运行时所花费的存储空间、内存消耗以及运行时间有所增加，这就需要在可靠性要求和额外付出代价之间作出折衷。

2. 检错技术

在软件系统中，无需在线容错的地方，或不能采用冗余设计技术的部分，如果对可靠性要求较高，故障有可能导致严重的后果，一般采用检错技术，在软件出现故障后能及时发现并报警，提醒维护人员进行处理。检错技术实现的代价一般低于容错技术和冗余技术，但它有一个明显的缺点，就是不能自动解决故障，出现故障后如果不进行人工干预，将最终导致软件系统不能正常运行。

采用检错设计技术要着重考虑几个要素：检测对象、检测延时、实现方式、处理方式。

- 检测对象：检测对象包含两个层次的含义，检测点和检测内容。在设计时应考虑把检测点放在容易出错的地方，和出错对软件系统影响较大的地方；检测内容选取那些有代表性的、易于判断的指标。
- 检测延时：从软件发生故障到被自检出来是有一定延时的，这段延时的长短对故障的处理是非常重要的，因此，在软件检错设计时要充分考虑到检测延时。如果延时长到影响故障的及时报警，则需要更换检测对象或检测方式。



- 实现方式：最直接的一种实现方式是判断返回结果，如果返回结果超出正常范围，则进行异常处理。计算运行时间也是一种常用的技术，如果某个模块或函数运行超过预期的时间，可以判断出现故障。另外，还有置状态标志位等多种方法，自检的实现方式要根据实际情况来选用。
- 处理方式：大多数检错采用“查出故障-停止软件系统运行-报警”的处理方式，但也有采用不停止或部分停止软件系统运行的情况，这一般由故障是否需要实时处理来决定。

3．降低复杂度设计

前面我们讲到，软件和硬件最大的区别之一就是软件的内部结构比硬件复杂得多，我们用软件复杂度来定量描述软件的复杂程度。软件复杂性常分为模块复杂性和结构复杂性。模块复杂性主要包含模块内部数据流向和程序长度两个方面，结构复杂性用不同模块之间的关联程度来表示。软件复杂度可用涉及到模块复杂性和结构复杂性的一些统计指标来进行定量描述，在这里就不进行详细叙述了。

软件的复杂性与软件可靠性有着密切的关系，软件复杂性是产生软件缺陷的重要根源，有研究表明，当软件的复杂度超过一定界限时，软件缺陷数会急剧上升，软件的可靠性急剧下降。因此，在设计的时候就应考虑降低软件的复杂性，使之处于一个合理的阈值之内，这是提高软件可靠性的有效方法。

降低复杂度设计的思想就是在保证实现软件功能的基础上，简化软件结构，缩短程序代码长度，优化软件数据流向，降低软件复杂度，从而提高软件可靠性。

除了容错设计、检错设计和降低复杂度设计技术外，人们尝试着把硬件可靠性设计中比较成熟的技术，如故障树分析（FTA）、失效模式与效应分析（FMEA）等运用到软件可靠性设计领域，这些技术大多是运用一些分析、预测技术，在软件设计时就充分考虑影响软件可靠性的因素，并采取一些措施地进行优化。由于软件与硬件内部性质的巨大差异，这些技术在软件可靠性设计领域的应用效果和范围极其有限。

15.5.2 软件可靠性管理

为了进一步提高软件可靠性，人们又提出了软件可靠性管理的概念，把软件可靠性活动贯穿于软件开发的全过程。

软件可靠性管理是软件工程管理的一部分，它以全面提高和保证软件可靠性为目标，以软件可靠性活动为主要对象，是把现代管理理论用于软件生命周期中的可靠性保障活动的一种管理形式。

软件可靠性管理的内容包括软件工程各个阶段的可靠性活动的目标、计划、进度、任务、修正措施等。

软件工程各个阶段可能进行的主要的软件可靠性活动如下所述。由于软件之间的差异较大，并且人们对可靠性的期望不同，对可靠性的投入不同，所以下面的每项活动并不是每一个软件系统的可靠性管理的必须内容，也不是软件可靠性管理的全部内容。

1．需求分析阶段

- 确定软件的可靠性目标；
- 分析可能影响可靠性的因素；
- 确定可靠性的验收标准；
- 制定可靠性管理框架；
- 制定可靠性文档编写规范；
- 制定可靠性活动初步计划；
- 确定可靠性数据收集规范。

2．概要设计阶段

- 确定可靠性度量；
- 制定详细的可靠性验收方案；
- 可靠性设计；
- 收集可靠性数据；
- 调整可靠性活动计划；
- 明确后续阶段的可靠性活动的详细计划；
- 编制可靠性文档。

3．详细设计阶段

- 可靠性设计；
- 可靠性预测（确定可靠性度量估计值）；
- 调整可靠性活动计划；
- 收集可靠性数据；
- 明确后续阶段的可靠性活动的详细计划；
- 编制可靠性文档。

4．编码阶段

- 可靠性测试（含于单元测试）；
- 排错；
- 调整可靠性活动计划；
- 收集可靠性数据；
- 明确后续阶段的可靠性活动的详细计划；
- 编制可靠性文档。



5．测试阶段

- 可靠性测试（含于集成测试、系统测试）；
- 排错；
- 可靠性建模；
- 可靠性评价；
- 调整可靠性活动计划；
- 收集可靠性数据；
- 明确后续阶段的可靠性活动的详细计划；
- 编制可靠性文档。

6．实施阶段

- 可靠性测试（含于验收测试）；
- 排错；
- 收集可靠性数据；
- 调整可靠性模型；
- 可靠性评价；
- 编制可靠性文档。

可靠性管理目前还停留在定性描述的水平上，很难用量化的指标来进行可靠性管理。可靠性管理规范的制定水平和实施效果也有待提高。怎样利用有限的可靠性投入，达到预期的可靠性目标是软件项目管理者常常要面对的难题。因此，可靠性管理研究是一个长期的课题。

第16章 文档测试

16.1 文档测试的范围

软件产品由可运行的程序、数据和文档组成。文档是软件的一个重要组成部分。

在软件的整个生命周期中，会用到许多文档，在各个阶段中以文档作为前阶段工作成果的体现和后阶段工作的依据。在软件的开发过程中，软件开发人员需根据工作计划和需求说明书由粗而细地进行设计，这些需求说明书和设计说明书构成了开发文档。为了使用户了解软件的使用、操作和对软件进行维护，软件开发人员需要为用户提供详细的资料，这些资料称为用户文档。而为了使管理人员及整个软件开发项目组了解软件开发项目安排、进度、资源使用和成果等，还需要制定和编写一些工作计划或工作报告，这些计划和报告构成了管理文档。软件文档的分类结构图如图16-1所示。

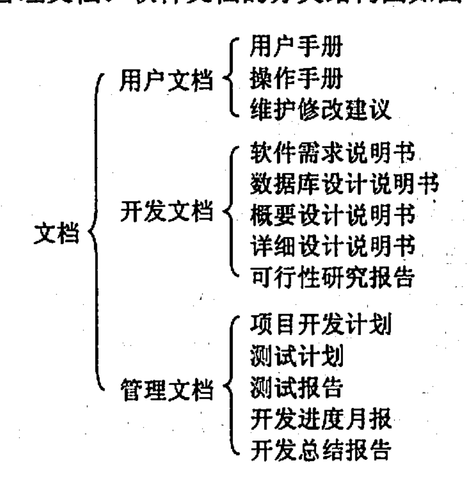


图16-1 软件的文档分类结构图

下面对这些文档进行一些说明。

- 可行性研究报告：说明该软件开发项目的实现在技术上、经济上和社会因素上的可行性，评述为了合理地达到开发目标可供选择的各种可能实施的方案，说明并论证所选定实施方案的理由。
- 项目开发计划：为软件项目实施方案制定出具体计划，应该包括各部分工作的



负责人员、开发的进度、开发经费的预算、所需的硬件及软件资源等。项目开发计划应提供给管理部门，并作为开发阶段评审的参考。

- 软件需求说明书：也称软件规格说明书，其中对所开发软件的功能、性能、用户界面及运行环境等作出详细的说明。它是用户与开发人员双方对软件需求取得共同理解的基础上达成的协议，也是实施开发工作的基础。
- 数据库设计说明书：只对使用数据库的软件适用，该说明书应给出数据库的整体架构及各个数据表中的逻辑关系。
- 概要设计说明书：该说明书是概要设计阶段的工作成果，它应说明功能分配、模块划分、程序的总体结构、输入输出以及接口设计、运行设计、数据结构设计和出错处理设计等，为详细设计奠定基础。
- 详细设计说明书：着重描述每一模块是怎样实现的，包括实现算法、逻辑流程等。
- 用户手册：本手册详细描述软件的功能、性能和用户界面，使用户了解如何使用该软件。
- 测试计划：计划应包括测试的内容、进度、条件、人员、测试用例的选取原则、测试结果允许的偏差范围等。
- 测试分析报告：测试工作完成以后，应提交测试计划执行情况的说明。对测试结果加以分析，并提出测试的结论意见。
- 开发进度月报：该月报是软件人员按月向管理部门提交的项目进展情况报告。报告应包括进度计划与实际执行情况的比较、阶段成果、遇到的问题和解决的办法以及下个月的打算等。
- 项目开发总结报告：软件项目开发完成以后，应与项目实施计划对照，总结实际执行的情况，如进度、成果、资源利用、成本和投入的人力。此外还需对开发工作作出评价，总结出经验和教训。
- 操作手册：本手册为操作人员提供该软件各种运行情况的有关知识，特别是操作方法的具体细节。
- 维护修改建议：软件产品投入运行以后，发现了需对其进行修正、更改等的问题，应将存在的问题、修改的考虑以及修改的估计影响作详细的描述，写成维护修改建议，提交审批。

以上这些文档是在软件生存期中，随着各阶段工作的开展适时编制的。其中有的仅反映一个阶段的工作，有的则需跨越多个阶段。

开发文档的测试在“软件生命周期测试策略”一章中已进行了详述，这里不再重复，本章以下部分将主要就用户文档的测试进行描述。

16.2 用户文档的内容

当用户文档仅包含一个 Readme 文件时，文档的测试只需要对其进行拼写检查，确认其中涉及到的技术准确无误，最多对 Readme 文件进行病毒扫描，确保其不带病毒就足够了。

但随着技术的进步和市场的规范，用户文档的范围越来越大了。以下这些都可以算是用户文档。并不是每一个软件都必须具有所有这些文档，但大多不出此列。

- 包装上的文字和图案。包括纸盒、包装纸或信封等。这些包装上可能含有软件的屏幕截图、特性清单、系统要求和版权信息等。
- 宣传材料、广告及其他插页。这些是软件开发者促进其相关软件销售的重要工作，同时提供补充内容、其他软件介绍和服务联系方式等。虽然对于一些用户来说，可能会随手丢弃，但同样也有一些用户会严肃地对待它们，因此这些信息也必须正确。
- 授权/注册登记表。这是希望用户填写内容、注册软件并寄回的卡片，它同样作为软件的一部分，也可能是电子文档，让用户在线阅读及注册。
- 最终用户许可协议。这是要求用户在使用软件前认可的一份法律文书，包括用户同意不得复制软件等内容。最终用户许可协议可能是打印在软盘或光盘的盒子上，或信封上，也有可能是在安装过程中弹出一个窗口显示在屏幕上。
- 标签和不干胶条。这类文档可能是软盘或光盘上的标签，或出现在包装盒上、印刷的材料上，像序列号、信封或光盘盒的封口标签等。这上面的内容也是需要测试的。
- 安装和设置指导。对于简单的软件来说，可能是包装中的一页纸，对于复杂软件来说，就有可能是完整的一本手册了。
- 用户手册。由于电子手册的实用性和灵活性，使纸介质的手册已大大减少了。目前一些软件附带简明的“入门”类小册子，而详细信息往往采用电子文档。电子手册大都以光盘形式随软件销售，也有的在网站上供用户下载。
- 联机帮助。联机帮助与用户手册有时可以互换使用，甚至取代用户手册。由于联机帮助可以有索引和搜索功能，一些联机帮助允许模糊查询或多关键字查询，更方便了用户查找所需信息。
- 指南、向导。这些工具已不仅仅是一页一页的文档，它们是文本内容和程序的结合，通常属于联机帮助的一部分。用户可以提出问题，然后向导将一步步引导用户完成任务。关于向导的例子可以参考微软 Office 的助手功能。



- 样例、示例和模板。字处理、网页制作等软件往往带有表单或样例，用户只需填写相应内容即可快速达到具有专业外观的效果。编译器可以用几段代码来演示如何使用编程语言的某些方面。财务软件可以通过模拟账套来解释软件的使用方法。
- 错误提示信息。这虽然是软件程序的一部分，但在程序测试中往往被忽略，而它们也是文档的一部分。

16.3 用户文档的作用

对于软件测试人员来说，对待用户文档要像对待程序一样给予同等关注和投入，因为对于用户来说，文档和程序同样重要。

充分有效的文档有如下优点。

- 改善易安装性。用户需要将软件产品安装到自己的计算机上。他们可能需要做复制文件、配置数据库、输入初始参数、将以前的数据导入等大量工作。安装程序是最后编写的，开发人员可能不会像对待产品的其他部分一样认真对待它，因为一些人认为用户仅会安装程序一两次而已，因此安装程序得到的测试和开发支持最少。然而，用户对产品的初次体验是从软件的安装开始的，如果在安装时遇到了困难，用户可能会对软件失去信心，或提出昂贵的技术支持要求，或干脆放弃使用软件。清晰、正确的安装指南是产品文档中最为重要的部分之一。
- 提高软件的易用性。具备优良文档的产品更易于使用。文档编制得越好，用户对产品的理解就越快，操作中发生的理解方面的错误就越少，效率就越高。高效的文档往往是面向任务的，它会估计用户意图，并说明如何完成各项任务。并不是说面向特征的用户手册就不好，面向特征的用户手册会独立地描述功能特征，按菜单顺序，甚至按字母顺序逐个描述功能。不同的软件类型适用于不同的手册类型，一些软件会同时提供这两类手册。
- 改善软件可靠性。不清晰、不正确的文档会降低产品的可靠性，用户使用它容易出现操作上的错误。优秀的文档即使在程序设计得很糟糕的情况下，也能有助于减少用户犯错次数。
- 促进销路。高质量的文档常会被作为卖点，可以帮助销售人员说明和推荐产品。在很多软件评审中它也扮演着重要的角色。
- 降低技术支持的费用。由用户发现问题比在产品开发早期发现问题的修复费用要高出数十倍。好的文档能够通过恰当的解释引导用户自己解决问题，尽可能

地避免用户打技术支持电话。如果用户文档描述了程序不具备的功能，开发商就是在作虚假的宣传；如果用户文档描述了实际上无法执行的操作，则开发商是在误导用户。不正确的指导会浪费用户不必要的时间和精力，也给开发商增加了法律上的风险。

16.4　用户文档测试需要注意的问题

对于软件用户来说，程序之外的部分也是软件的一部分，他们并不管这些东西是由程序员、作家还是图形艺术家创建的。他们关心的是整个软件包的质量。

文档测试中需要注意如下问题。

- 文档常常得不到足够的重视，文档的开发缺乏足够的资金和技术支持，而文档的测试更得不到重视。一个好的软件项目，一定要为文档测试留出预算，像对程序一样对文档给予关注。对文档中发现的缺陷，也需像发现程序缺陷一样给出报告。
- 编写文档的人可能并不是软件特性方面的专家，对软件功能可能了解得并不深入。其结果就是写出来的产品说明书可能并不到位，或者不能解释复杂的产品特性。软件文档测试人员可以与文档作者紧密合作，保证文档中所包含信息的质量，并随着产品的更新而更新。更重要的是，测试人员可以发现并指出程序中难以使用或难以理解之处，让文档作者在文档中作出更好的解释。
- 由于文档的印刷需要花费不少的时间，可能是几周，如果追求印刷质量的话可能需要几个月。而在这段时间内，软件发现的错误可以有时间修改，程序很可能已经发生了改变，而文档无法反映最终的修改。Readme 文件的发明正因为如此，它是将最后的改动通知用户的方式。它能使文档保持到最后一刻发布，从而保证与软件程序的同步。同时，随软件发布的联机帮助信息也可以尽可能地反映最新的修改。如果文档测试不够充分，大量的错误将不得不随着印刷精美的手册到达用户手中，而 Readme 文件就不是最新特性的发布而是长长的勘误表了。
- 文档测试不仅仅是对文字的校对，更可以辅助找到更多的程序错误。文档编写人员与文档测试人员审视程序的角度与程序员和程序测试人员并不相同，因此由文档测试揭示的问题也不同于程序员和程序测试人员所发现的问题，文档测试往往会发现其他测试无法发现的严重错误，例如，功能实现错误、易用性不好、用户手册与程序实现不吻合等问题。当然，这是在全面测试的基础上，而全面测试意味着每 3~5 页花费 1 小时的时间。测试人员审看文档的速度越快，从文档和程序中发现问题的机会就越少。加强测试监督、重新培训测试人员，

甚至更换测试人员能有助于解决这一问题。

16.5 用户文档测试的要点

文档测试分为两类，如果是非程序，例如打印的手册或产品包装盒，其测试可以视为技术校对。如果文档和程序紧密结合，例如超链接形式的电子手册或联机帮助，或助手一类的帮助系统，就要进行与程序测试类似的测试。

无论是文档或是程序，作为文档测试人员，都必须像用户那样对待它，应该说像最仔细的用户那样，认真阅读，跟随每个步骤，检查每个图形，尝试每个示例。只有这样，才能尽可能找出软件和文档中的缺陷。

文档测试中，对于如下几个方面需要特别关注。

- 读者群。文档面向的读者定位要明确。对于初级用户，可能需要从鼠标的用法、点击确定按钮等讲起；对于中级用户，重要界面的截图和关键步骤每一个参数的选择方法都需要介绍；对于高级用户，则没有必要给出太多的界面截图，但对重要参数的讲解一定要深入，用词要专业。特别是不论用户群定位如何，文档都不可以写成散文、诗歌或者侦探、言情小说，文档的目的是要让用户看得懂，能理解。
- 术语。文档中用到的术语要适用于定位的读者群，用法一致，标准定义与业界规范相吻合。如果有索引或交叉引用，所有的术语都应能够进行索引和交叉引用。如果术语较多，在纸介质手册的末尾应给出术语索引；如果被测软件提供二次开发功能，有大量函数，则有必要编写独立的函数手册和开发指南。
- 正确性。这是非常重要的，会占用文档测试的大量时间和人力。测试中需检查所有信息是否真实正确，查找由于过期产品说明书和销售人员夸大事实而导致的错误。检查所有的目录、索引和章节引用是否已更新，尝试链接是否准确，产品支持电话、地址和邮政编码是否正确。
- 完整性。慢慢地仔细阅读文字，完全根据提示进行操作，不要作任何假设。对照软件界面检查是否有重要的分支没有描述到，甚至是否有整个大模块没有描述到，耐心补充遗漏的步骤。用户不会知道遗漏了什么信息，直到有一天他使用软件时走到了这个分支。对于极其熟悉被测软件的人来说，这项测试相当困难，因为思路已固定地按照一定的流程去测试，极易忽略不常用的部分。因此，可以考虑让不是很熟悉被测软件的人员进行此项目的测试。
- 一致性。按照文档描述的操作执行后，检查软件返回的结果是否与文档描述相同。要留意软件界面上出现的版本号与手册、帮助上的信息是否一致。

- 易用性。纸介质文档可以通过目录、关键词索引提高用户使用的易用性。条理清晰、结构合理的文档是优质软件的一个显著特征。对关键步骤以粗体或背景色给用户以提示，合理的页面布局、适量的图表都可以给用户更高的易用性。电子文档或帮助系统显然比纸介质在这方面有更大的优势。需要注意的是，文档要有助于用户排除错误，只描述正确操作而不描述错误处理办法的文档是不负责任的。与程序大多用于错误处理一样，文档对于用户看到的错误信息应当有更详细的文档解释，而且不应让用户花费太多的时间去寻找所需的解释。
- 图表与界面截图。检查所有图表与界面截图是否与发行版本相同。对于成熟的软件开发商来说，界面在设计阶段就应基本确定，不应在软件开发后期有大的变动。而此项测试就是要发现在文档完成后是否有界面变动，确保屏幕截图源于发行版本。测试中还要注意图表标题的正确性。
- 样例和示例。像用户一样载入和使用样例。如果是一段程序，就输入数据并执行它。以每一个模板制作文件，确认它们的正确性。想像一下样例不能执行的问题交给技术支持人员时的情景……
- 语言。对于英语文档来说，拼写和语法检查器太常见了，一般不会出现拼写和语法错误。但对于中文文档来说，可以采用一些校对工具辅助人工检查，并进行细致专业的校对，不要让用户发现错别字，不要出现有二义性的说法。特别要注意的是屏幕截图或绘制图形中的文字，不能想当然，没有任何工具能够从图形中找出语法错误。
- 印刷与包装。文档测试似乎完成了，文档终于变成精美的册子，这时，测试人员还需抽查印刷质量，看看手册厚度与开本是否合适，翻看起来是否方便，包装盒的大小是否合适，光盘盒的固定有没有问题，有没有零碎易丢失的小部件等等。这时发现的问题，如果不是太严重，已不可能在这个版本中进行改进，但对下一个版本的制作来说是非常有价值的。

16.6 针对用户手册的测试

用户手册是用户文档中最重要的一部分。在对用户手册进行测试时，应拿着它坐在计算机前，进行如下操作。

- 准确地按照手册的描述使用程序。在每个例子中如实地进行每个键盘操作。用户在按照手册运行程序时可能会进行错误的操作，因此测试时测试人员也可以随心所欲地“犯错误”。检查计算机对错误的处理和手册对错误处理的描述应当占用测试人员的大部分精力。



- 尝试每一条建议。即使建议并没有完全表达清楚，仍应按步骤去尝试。用户依照建议会做什么，测试人员就应当做什么，甚至尝试更多的可能性。
- 检查每条陈述。测试人员需要对每条陈述进行检查，因为用户手册是产品最终的规范，是用户检查程序运行是否正确首先求证的地方。
- 查找容易误导用户的内容。有些示例和特征描述得并不准确，一般的读者可能会从中归纳出错误的结论。用户可能对程序的能力抱有过高的期望，或是凭空想象一些实际并不存在的约束条件。尽早标识出易被人误解的内容，这一点极其重要。

16.7 针对在线帮助的测试

帮助文档的测试在很大程度上与用户手册测试相同，但帮助并不只是用户手册的电子版，因此再给出以下几点补充说明。

- 准确性。对帮助准确性检查的细致程度至少要接近于对用户手册的检查。通常帮助文本都没有得到良好的处理和充分测试，无法受到用户的欣赏。一旦用户发现帮助中存在明显的错误，他可能对帮助系统的信任程度大大下降。
- 帮助是文档编写和程序编写的结合。不仅要检查文本的准确性，还要检查程序的可靠性。制作帮助的人员往往并不是专业的程序员，他们在使用帮助制作工具的技巧、与程序的接口等问题上不可能达到完美。
- 帮助索引。如果帮助系统包含了索引或主题列表，允许用户由索引进入到主题中，测试人员就必须逐条进行检查。
- 超链接。除非是早期开发的软件或开发商对帮助过于不重视，超链接是在线帮助中必须的功能。测试人员必须对每个链接都测试到，复杂的超链接可能会对一个主题形成树状结构展开的若干页面，甚至构成网状结构。测试人员有义务检查每条分支。
- 链接的意义。索引和链接的条目应当是有意义的，测试人员需要发现是否有一些帮助主题未出现在索引里，或出现的名称不恰当。如果用户不能迅速找到所需的信息，只能说帮助系统在一程度上是失败的。
- 帮助的风格。很少会有用户能悠闲地查看帮助，帮助的阅读者是带着问题、焦躁不安而缺乏耐心的。帮助文本需要比用户手册更为简洁，风格也应更为简单。良好的帮助系统应该是面向任务或面向操作的，它必须提供一些有意义的信息，让用户能立即开始或继续他的操作。任何在帮助中出现的令人迷惑或离题的内容都可以作为测试问题。

第三篇 测试案例

第17章 功能测试

17.1 概述

前面我们介绍了多种黑盒测试设计的方法，比如等价类划分法、因果图法、错误推测法、场景法等，这些方法在测试过程中如何选择及应用，需要根据测试项目的特点，结合测试经验灵活使用。为了便于读者在较短的时间内确实掌握上述方法的使用，我们在本章中以业务流程较复杂、功能点较多、集成性较高的ERP软件测试为实例，在中国软件评测中心长期测试经验积累的基础上，对黑盒测试的各种方法作一个深入的介绍，力争使读者对黑盒测试的实施过程有一个全面的认识。

17.2 ERP软件简介

企业资源计划（Enterprise Resource Planning，ERP）即ERP企业资源计划是一种先进的企业管理理念，它将企业各个方面的资源进行充分地调配和平衡，为企业提供多重解决方案，使企业在激烈的市场竞争中取得竞争优势。ERP以制造资源计划MRPII为核心，基于计算机技术的发展，并进一步吸收了现代的管理思想。ERP主要侧重于对企业内部人、财、物等资源的管理，并且扩展了管理范围，它把企业需求和制造活动以及供应商的制造资源整合在一起，形成了一个完整的供应链，并且将供应链上所有环节如定单、采购、库存、计划、生产、发货和财务等所需的所有资源进行统一的计划和管理。ERP软件特点是业务流、数据流、资金流、管理流集成化程度高，并且各模块联系紧密。其主要功能包括生产制造控制、物流控制、财务管理、人力资源管理、设备管理、质量管理等、库存管理等。

17.3 ERP软件测试的难点

ERP软件是一种流程复杂、功能点多且关联性强的系统。如果按照对一般应用软件的测试方法进行测试，即使耗费很大的人力、物力进行测试，保证大部分功能点都正确，



也不能保证可以正常地使用，因为ERP软件的业务流顺畅、集成性高是更重要的要求。针对这样的难点，我们将测试重点应该放在流程正确集成上。

测试ERP软件，要求测试人员不仅要掌握ERP业务流程和ERP管理思想，还要了解行业及企业的需求。在项目实施过程中要求测试工程师协同工作，共同来设计ERP软件的测试用例，并进行测试。

这里我们提出以业务流和数据流为主驱动的方法设计用例。

17.4 ERP软件测试实例及分析

本实例以适用于离散制造业、面向定单的生产方式的一类ERP软件为例，对其基础数据模块、销售管理模块、计划管理模块、采购管理模块、生产管理模块的主要功能和基本流程测试进行介绍。实例模拟了销售部门签订销售定单，之后转到计划部门对销售定单进行物料需求计算，采购部门和生产部门根据计划部门下达的计划进行生产和采购，最终完成发货并关闭销售定单的基本流程。该实例采用流程图的方式，侧重于业务流、数据流、资金流以及管理流的测试。

用例设计首先使用场景法，对系统运行流程进行分析，从宏观考虑用例应包括的哪些基本流和被选流，其次在设计具体的数据流时以业务流为驱动，结合等价类划分、边界值分析、因果图等方法进行具体数据的设计。

在进行用例设计时，需要强调必须依据软件设计说明书和使用手册进行操作。

17.4.1 前期分析

由于ERP软件的流程比较复杂，如何选择有限的有代表性的流程达到测试需求，在设计测试用例前，利用场景法对软件的流程进行分析，通过用例场景并结合各路径的触发条件来确定用例应遵从的流程。

所谓用例场景，就是在测试用例设计方法中介绍过的，通过描述流经用例的路径来确定测试用例的过程，这个流经路径要从用例开始到结束，遍历其中所有基本流和备选流。

1．业务流程图

如图17-1所示是我们用例中主要模块的业务流程图。

2．主备选流图

根据上面的流程图和用户使用手册，我们可以归纳出一个看上去比较清晰的主、备选流关系图，如图17-2所示以及各路径与触发条件的对照表，如表17-1所示。

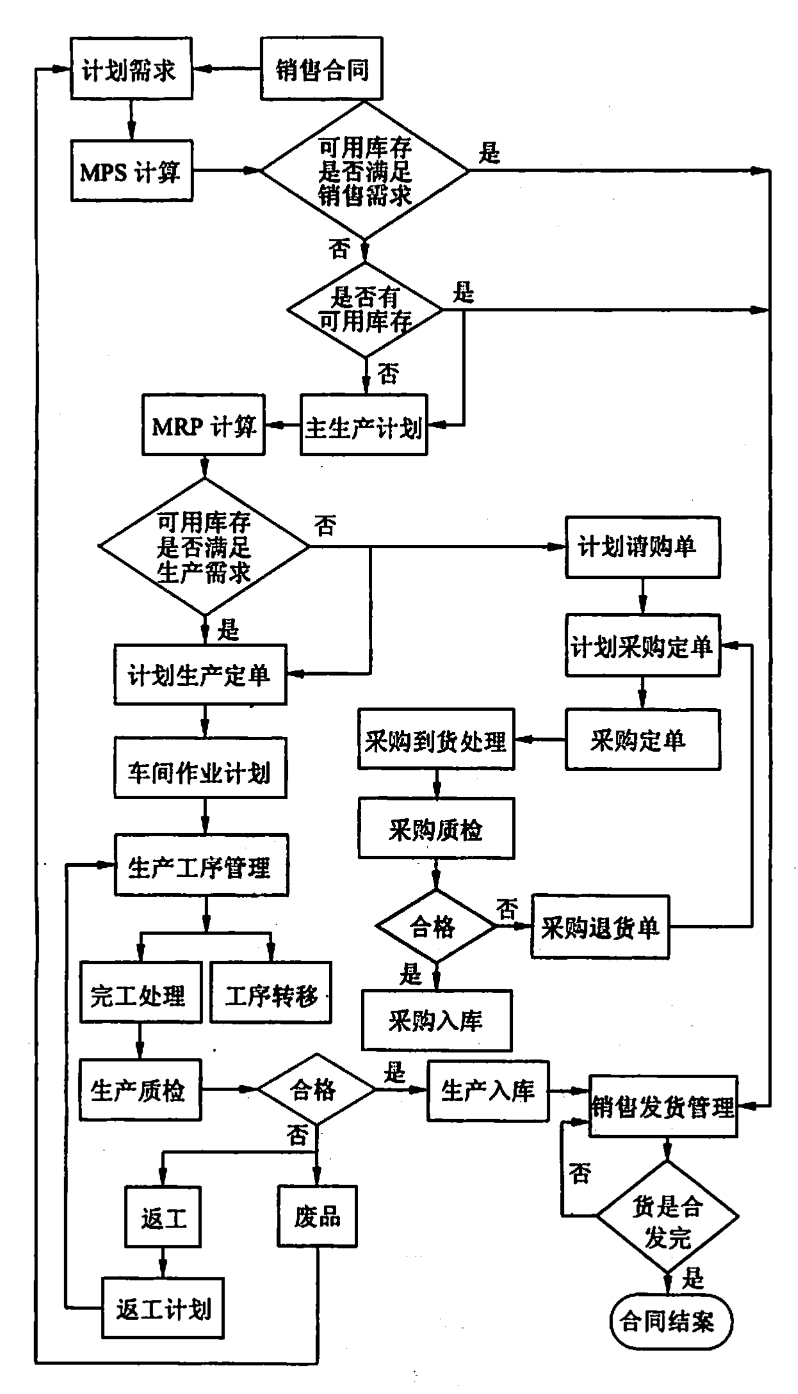


图 17-1 主要模块的业务流程图

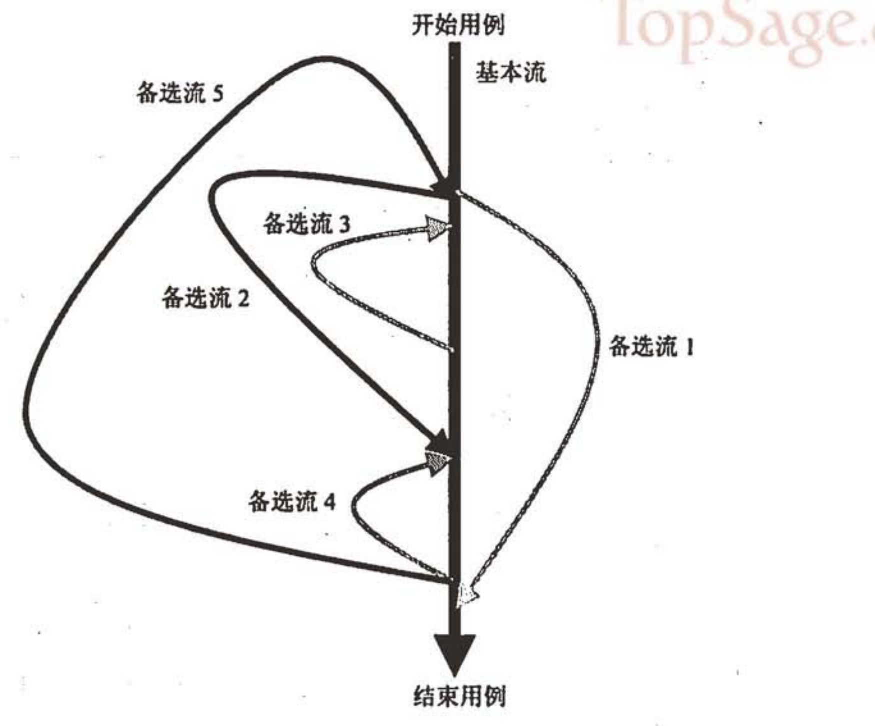


图 17-2 主、备选流关系图

表 17-1 各路径的触发条件对照表

路　径	触发条件
基本流	① 库存可用产品数量不满足销售需求 ② 库存可用零部件数量不能满足生产需求 ③ 所采购的部件入库质检全部合格 ④ 所生产的部件及产品全部合格
备选流 1	库存可用产品数量满足销售需求
备选流 2	① 库存可用产品数量不满足销售需求 ② 库存可用零部件数量能满足生产需求
备选流 3	① 库存可用产品数量不满足销售需求 ② 库存可用零部件数量不能满足生产需求 ③ 所采购的部件入库质检部分不合格
备选流 4	① 库存可用产品数量不满足销售需求 ② 所生产的部件及产品需返工
备选流 5	① 库存可用产品数量不满足销售需求 ② 所生产的部件及产品有废品

3．场景分析

遵循如图 17-2 中所示的路径，可以确定不同的用例场景，从基本流开始，将基本流和备选流结合起来，可以确定各种场景（如表 17-2 所示只是列出部分的场景）。

表 17-2　场景路径表

场景 1	基本流
场景 2	基本流：备选流 1
场景 3	基本流：备选流 2
场景 4	基本流：备选流 3
场景 5	基本流：备选流 4
场景 6	基本流：备选流 5
场景 7	基本流：备选流 2：备选流 4
场景 8	基本流：备选流 3：备选流 4
场景 9	基本流：备选流 5：备选流 1
场景 10	基本流：备选流 2：备选流 5
场景 11	基本流：备选流 3：备选流 5
场景 12	基本流：备选流 5：备选流 4
场景 13	基本流：备选流 5：备选流 2：备选流 4
场景 14	基本流：备选流 5：备选流 3：备选流 4
场景 15	基本流：备选流 2：备选流 4：备选流 5：备选流 3

以上我们讨论了 ERP 几个子模块之间的业务流程图，同时模块内部还有较复杂的业务流程，在实际测试时我们不可能对所有流程一一验证，这就引出一个问题：如何选择“性价比”较高的业务流程，使它们尽量覆盖较多的场景，然后根据所选业务流设计数据流。为了解决各这个问题，我们建立了路径触发条件与场景关系表，如表 17-3 所示。

表 17-3　路径触发条件与场景关系表

序　号	路径触发条件组合	覆盖的场景
1	① 库存无可用产品数量销售需求 ② 库存无可用零部件 ③ 所采购的部件入库质检全部合格 ④ 所生产的部件及产品全部合格	场景 1
2	库存可用产品数量满足销售需求	场景 2
3	① 库存中有可用产品但不满足销售需求 ② 库存无可用零部件 ③ 所采购的部件入库质检全部合格 ④ 所生产的部件及产品全部合格	场景 1、场景 2



续表

序　号	路径触发条件组合	覆盖的场景
4	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检全部合格 ④ 所生产的部件及产品全部合格	场景 1、场景 2、场景 3
5	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检全部不合格 ④ 所生产的部件及产品全部合格	场景 2、场景 3、场景 4
6	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检部分不合格 ④ 所生产的部件及产品全部合格	场景 1、场景 2、场景 3、场景 4
7	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检部分不合格 ④ 所生产的部件及产品全部返修	场景 2、场景 5、场景 7、场景 8
8	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检部分不合格 ④ 所生产的部件及产品全部为废品	场景 2、场景 6、场景 10、场景 11
9	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检部分不合格 ④ 所生产的部件及产品中部分为废品，其余部分需要返修	场景 2、场景 5、场景 6、场景 7、场景 8、场景 10、场景 11
10	① 库存中有可用产品但不满足销售需求 ② 库存有可用零部件但不满足生产需求 ③ 所采购的部件入库质检部分不合格 ④ 所生产的部件及产品中部分为废品、部分需要返修；部分合格	场景 1、场景 2、场景 3、场景 4、场景 5、场景 6、场景 7、场景 8、场景 10、场景 11

分析：从表 17-3 中可以看出第 10 组条件组合所覆盖的场景最多，应该按照这个组合设计案例（实际测试中可以根据软件需求和测试需求的不同，添加或减少触发条件），但其同时存在着优点和缺点。

缺点：对循环执行业务考虑得不全，如未覆盖场景 9、12、13、14、15，归其原因

是在于没有考虑执行备选流 5 以后的场景触发条件。

优点：覆盖了全部流程分支，且可以按照实际测试需求，根据这个条件组合循环执行案例，达到要求的场景覆盖率。

通过以上工作我们确定了在设计该 ERP 软件案例时“性价比”较高的流程，以及触发流程所需的基本条件，这样在准备案例的数据流时就有了“根基”，使一套测试数据能够覆盖尽量多的流程分支以及功能点。反之，如果盲目地选择流程进行案例设计，结果可能是重要的流程分支及功能点没有覆盖到，或者是虽然流程分支及功能点覆盖到了，但进行了大量的重复性劳动，造成了人力、物力的浪费。

下面我们就以表 17-3 中第 10 组条件组合为例，进行案例设计。

17.4.2　用例设计

1．基础数据管理

基础数据模块是系统的基础，多数功能为基础数据管理，因此该模块不列举流程图。在本部分，我们以会计期间输入测试用例的设计为例，介绍两种测试用例设计方法：等价类划分法和边界值法。

（1）会计期间输入测试用例设计

如表 17-4 所示是一个已设置完成的会计期间，包括三个字段，其中“序号”由软件自动生成的，“起始日期”和“终止日期”手工录入。

表 17-4　会计期间

序　号	起始日期	终止日期	序　号	起始日期	终止日期
1	2003-01-01	2003-01-31	7	2003-07-01	2003-07-31
2	2003-02-01	2003-02-28	8	2003-08-01	2003-08-31
3	2003-03-01	2003-03-31	9	2003-09-01	2003-09-30
4	2003-04-01	2003-04-30	10	2003-10-01	2003-10-31
5	2003-05-01	2003-05-31	11	2003-11-01	2003-11-30
6	2003-06-01	2003-06-30	12	2003-12-01	2003-12-31

首先我们用到等价类划分法。等价类划分是把程序的输入域划分成若干部分，然后从每个部分中选取少数代表性数据作为测试用例。每一类的代表性数据在测试中的作用等价于这一类中的其他值，也就是说，如果某一类中的一个例子发现了错误，这一等价类中的其他例子也能发现同样的错误；反之，如果某一类中的一个例子没有发现错误，则认为这一类中的其他例子也不存在错误。

- 分析会计期间的要求和特点。



① 不能有无效的日期。比如：2003-02-29。

② 相邻时间段之间不能有日期间隔。

③ 时间段之间不能有日期的交集。

④ 必须按日期格式录入。比如：yyyy-mm-dd、mm/dd/yyyy。

⑤ 不能包含非法字符（yyyy、mm、dd 均为大于 0 的整数）。比如：字母、特殊字符等。

⑥ 起始日期或终止日期不能为空。

⑦ 不能存在无效时间段。如表 17-5 所示的时间段即为无效时间段。

表 17-5 无效时间段示例表

起始日期	终止日期
2003-03-31	2003-03-01

- 列出等价类列表，如表 17-6 所示。

表 17-6 等价类列表

条件	有效等价类	无效等价类
是否有不存在的日期	有效的日期 （1）	不存在的日期 （2）
相邻时间段是否有日期间隔	无日期间隔 （3）	有日期间隔 （4）
时间段之间是否有日期的交集	无日期交集 （5）	有日期交集 （6）
是否符合日期格式要求	yyyy-mm-dd （7） mm/dd/yyyy （8）	日期格式不是 yyyy-mm-dd 且不是 mm/dd/yyyy （9）
是否包含非法字符	yyyy>0 （10） mm>0 （11） dd>0 （12） yyyy 是整数 （13） mm 是整数 （14） dd 是整数 （15）	yyyy≤0 （16） mm≤0 （17） dd≤0 （18） yyyy 中含字母 （19） yyyy 中含特殊符号 （20） yyyy 为小数 （21） mm 中含字母 （22） mm 中含特殊符号 （23） mm 为小数 （24） dd 中含字母 （25） dd 中含特殊符号 （26） dd 为小数 （27）
起始日期或终止日期是否为空	起始日期不为空且终止日期不为空 （28）	起始日期为空 （29） 终止日期为空 （30）
是否存在无效时间段	起始日期早于终止日期 （31）	起始日期晚于终止日期 （32） 起始日期等于终止日期 （33）

- 会计期间输入的测试用例，如表 17-7 所示。

表 17-7　会计期间测试用例

序号	数　据	覆盖等价类	预期结果	实际输出
1	起始日期　终止日期 2003-01-01　01/31/2003 2003-02-01　02/28/2003 03/01/2003　03/31/2003	1、3、5、7、8、10、11、12、13、14、15、28、31	保存成功	
2	起始日期　终止日期 2003-01-01　2003-01-31 2003-02-03　2003-02-28	4、10、11、12、13、14、15、28、31	提示："录入会计期间存在日期间隔"，保存不成功	
3	起始日期　终止日期 2003-01-01　2003-01-31 2003-01-30　2003-02-28	6、10、11、12、13、14、15、28、31	提示："录入会计期间存在日期交集"，保存不成功	
4	起始日期　终止日期 2003-01-32　2003-02-29	2	提示："2003-01-32、2003-02-29 日期不存在"，保存不成功	
5	起始日期　终止日期 2003\01\01　2003.01.31	9	提示："2003\01\01、2003.1.31 日期格式不正确"，保存不成功	
6	起始日期　终止日期 -2003-01-01　01/31/2003 起始日期　终止日期 2003-01-01　01/31/-2003	16	提示："日期中包含非法字符"，保存不成功	
7	起始日期　终止日期 01/-01/2003　01/31/2003 起始日期　终止日期 01/01/2003　01/-31/2003	17	提示："日期中包含非法字符"，保存不成功	

续表

<table>
<tr><th>序号</th><th>数　　据</th><th>覆盖等价类</th><th>预 期 结 果</th><th>实际输出</th></tr>
<tr><td>8</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>-01/01/2003</td><td>01/31/2003</td></tr></table>
<table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>01/01/2003</td><td>-01/31/2003</td></tr></table></td><td>18</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
<tr><td>9</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>ab03-01-01</td><td>2003-01-31</td></tr></table>
<table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>ab03-02-28</td></tr></table></td><td>19</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
<tr><td>10</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-a1-01</td><td>2003-01-31</td></tr></table>
<table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-a2-28</td></tr></table></td><td>22</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
<tr><td>11</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-a1</td><td>2003-01-31</td></tr></table>
<table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-02-a8</td></tr></table></td><td>25</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
<tr><td>12</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2$03-01-01</td><td>2003-01-31</td></tr></table>
<table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2$03-02-08</td></tr></table></td><td>20</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
<tr><td>13</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-@1-01</td><td>2003-01-31</td></tr></table></td><td>23</td><td>提示：“日期中包含非法字符”，保存不成功</td><td></td></tr>
</table>

续表

<table>
<tr><th>序号</th><th>数　据</th><th>覆盖等价类</th><th>预 期 结 果</th><th>实际输出</th></tr>
<tr><td>13</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-@2-08</td></tr></table></td><td>23</td><td></td><td></td></tr>
<tr><td>14</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-*1</td><td>2003-01-31</td></tr></table><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-02-*8</td></tr></table></td><td>26</td><td>提示："日期中包含非法字符"，保存不成功</td><td></td></tr>
<tr><td>15</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>0.2003-01-01</td><td>2003-01-31</td></tr></table><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>0.2003-02-08</td></tr></table></td><td>21</td><td>提示："日期中包含非法字符"，保存不成功</td><td></td></tr>
<tr><td>16</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-0.1-01</td><td>2003-01-31</td></tr></table><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-0.2-08</td></tr></table></td><td>24</td><td>提示："日期中包含非法字符"，保存不成功</td><td></td></tr>
<tr><td>17</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-0.1</td><td>2003-01-31</td></tr></table><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-02-0.8</td></tr></table></td><td>27</td><td>提示："日期中包含非法字符"，保存不成功</td><td></td></tr>
<tr><td>18</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td></td><td>2003-02-28</td></tr></table></td><td>29</td><td>提示："起始日期为空"，保存不成功</td><td></td></tr>
<tr><td>19</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-01</td><td></td></tr></table></td><td>30</td><td>提示："终止日期为空"，保存不成功</td><td></td></tr>
</table>

续表

<table>
<tr><th>序号</th><th>数　　据</th><th>覆盖等价类</th><th>预 期 结 果</th><th>实际输出</th></tr>
<tr><td>20</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-31</td><td>2003-01-01</td></tr></table></td><td>32</td><td>提示：“起始日期晚于终止日期”，保存不成功</td><td></td></tr>
<tr><td>21</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-01</td><td>2003-01-01</td></tr></table></td><td>33</td><td>提示：“起始日期等于终止日期”，保存不成功</td><td></td></tr>
</table>

这个测试用例已经覆盖了全部等价类，但对具体输入数据的测试还不够完善，在此我们引入边界值分析法。

边界值分析法不是选择等价类的任意元素，而是选择等价类边界构建测试用例，是对等价类划分的很好的补充。实践证明，在设计测试用例时，对边界附近的处理必须给予足够的重视，为检验边界附近的处理专门设计测试用例，常常取得良好的测试效果。利用边界值分析法设计用例，可以有效地弥补等价类划分对具体用例数据设计上的不足。

首先检查存在的边界值：

① 日期的边界值，如 2003-01-01、2003-01-31、2004-02-29。

② 相邻会计期间的边界值。

③ 跨年度的时间段。

由此设计相应的测试用例如表 17-8 所示。

表 17-8　边界值测试用例

<table>
<tr><th>序　号</th><th>数　　据</th><th>期 望 输 出</th><th>实际输出</th></tr>
<tr><td>1</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-01</td><td>01/31/2003</td></tr></table></td><td>保存成功</td><td></td></tr>
<tr><td>2</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-02-01</td><td>2003-02-29</td></tr></table></td><td>提示：“2003-02-29 日期不存在”，保存不成功</td><td></td></tr>
<tr><td>3</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2004-02-01</td><td>2004-02-29</td></tr></table></td><td>保存成功</td><td></td></tr>
<tr><td>4</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-01-01</td><td>2003-01-31</td></tr><tr><td>2003-01-31</td><td>2003-02-28</td></tr></table></td><td>提示：“录入会计期间存在日期交集”，保存不成功</td><td></td></tr>
<tr><td>5</td><td><table><tr><td>起始日期</td><td>终止日期</td></tr><tr><td>2003-12-01</td><td>2004-01-01</td></tr></table></td><td>保存成功</td><td></td></tr>
</table>

（2）基础数据

说明：以下列出本实例中所用到的基础数据，针对它们的输入测试用例可以仿照上面介绍的方法进行设计，此处不再一一列出。

① 如表 17-9 所示为货币类别。

表 17-9　货币类别

编　码	名　称	备　注
RMB	人民币	本位币
USD	美元	

② 如表 17-10 所示为部门员工信息。

表 17-10　部门、员工信息

编　码	名　称	员　工	编　码	名　称	员　工
001	采购部		00301	总装车间	总装工程师
00101	采购一部	采购员甲	00302	加工车间	加工工程师
00102	采购二部	采购员乙	00303	外协加工	外协人员
002	销售部		004	仓储部	
00201	销售一部	销售员甲	00401	原材料库	库管员甲
00202	销售二部	销售员乙	00402	半成品	库管员乙
003	生产车间		00403	成品	库管员丙

③ 如表 17-11 所示为会计科目（参照国家财务制度会计科目设置标准）

表 17-11　会计项目

科目代码	科目名称	借贷方向	备注	科目代码	科目名称	借贷方向	备注
1101	现金	借		4101	生产成本	借	
1111	存款	借		4201	制造费用	借	
1141	应收账款	借		5101	销售收入	贷	
2121	应付账款	贷		5111	销售税金	借	

④ 如表 17-12 所示为仓库类别。

表 17-12　仓库类别

编　码	名　称	会计核算期	编　码	名　称	会计核算期
YCLK	原材料库	1	CPK	成品库	1
BCPK	半成品库	1	WXYLK	外协原料库	1

⑤ 如表 17-13 所示为产成品物料表。



表 17-13 产成品物料表 （价格单位：元）

名 称	来 源	类 型	编 码	计量单位	产品价格	生产提前期（天）	批 量
产成品	制造		10				
家用电脑	制造	产成品	10-C001	台	5000	1	5

如表 17-14 所示为半成品物料表。

表 17-14 半成品物料表 （价格单位：元）

名 称	来 源	类 型	编 码	计量单位	生产提前期（天）	批 量
半成品	制造		20			
主机	制造	半成品	20-C001	台	1	5

如表 17-15 所示为原材料物料表。

表 17-15 原材料物料表 （价格单位：元）

名 称	来 源	类 型	编 码	计量单位	标准价	采购提前期（天）	批 量
原材料	采购		30				
键盘	采购	原材料	30-C001	个	100	3	10
CPU	采购	原材料	30-C002	个	1000	3	10
硬盘	采购	原材料	30-C003	个	1000	3	10
内存	采购	原材料	30-C004	个	100	3	10
鼠标	采购	原材料	30-C005	个	50	3	10
主板	采购	原材料	30-C006	个	300	3	10
显示器	采购	原材料	.30-C007	台	1000	3	10
机箱	采购	原材料	30-C008	个	350	3	10

⑥ 如表 17-16 所示为物料企业供应商报价。

表 17-16 物料企业供应商报价 （价格单位：元）

物料编码	物料名称	价格类别	计量单位	单价	币种	生效时间	价格失效时间
30-C001	键盘	供应商价	个	100	人民币	2002-12-10	2003-12-10
30-C002	CPU	供应商价	个	1000	人民币	2002-12-10	2003-12-10
30-C003	硬盘	供应商价	个	1000	人民币	2002-12-10	2003-12-10
30-C004	内存	供应商价	个	100	人民币	2002-12-109	2003-12-10
30-C005	鼠标	供应商价	个	50	人民币	2002-12-10	2003-12-10
30-C006	主板	供应商价	个	300	人民币	2002-12-10	2003-12-10
30-C007	显示器	供应商价	个	1000	人民币	2002-12-10	2003-12-10
30-C008	机箱	供应商价	个	350	人民币	2002-12-10	2003-12-10

⑦ 如表 17-17 所示为往来客户资料。

表 17-17　往来客户资料

往来客户代码	往来客户简称	往来客户名称	客户标志	供应商标志
0001	供应商 A	供应商 A		G
0002	供应商 B	供应商 B		G
0003	客户 A	客户 A	K	
0004	客户 B	客户 B	K	

⑧ 如表 17-18 所示为计量单位。

表 17-18　计量单位

计量单位代码	计量单位简称	计量单位名称	计量单位说明
001	台	台	
002	个	个	

⑨ 如图 17-3 所示为产品结构图。

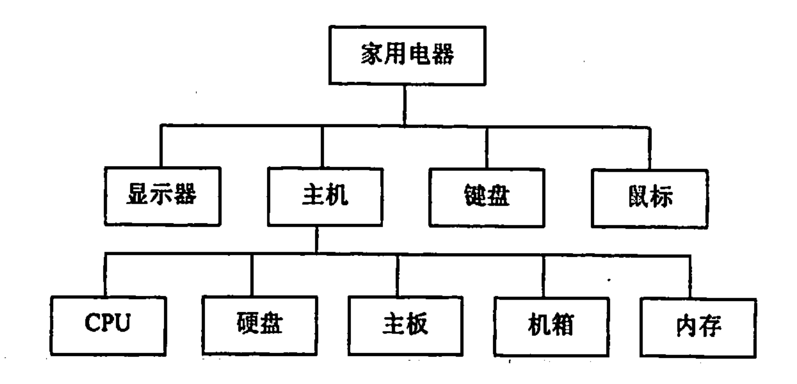


图 17-3　产品结构图

⑩ 如表 17-19 所示为工作中心定义表。

表 17-19　工作中心定义

工作中心代码	工作中心名称	所属车间	工作中心类别	人员数	关键工作中心
10001	总装一线	总装车间	按人员	5	否
20001	加工一线	加工车间	按人员	5	是

⑪ 如表 17-20 所示为工序名称。



表 17-20 工序名称

工序代码	工序名称
001	总装
002	加工

⑫ 如表 17-21 所示为工艺路线。

表 17-21 工艺路线

物料代码	工艺路线代码	工序	工序名称	工作中心	加工工时	批量	等待工时	准备工时	第一道工序	最后一道工序	是否质检
10-C001	1	20	总装	总装一线	8 小时	5	0	0	否	是	是
20-C001	1	10	加工	加工一线	8 小时	5	0	0	是	否	否

⑬ 如表 17-22 所示为企业日历

表 17-22 企业日历

日历码	日期	社会星期	企业星期	状态
…	…	…	…	…
01	2003-1-13	星期一	星期一	工作
01	2003-1-14	星期二	星期二	工作
01	2003-1-15	星期三	星期三	工作
01	2003-1-16	星期四	星期四	工作
01	2003-1-17	星期五	星期五	工作
01	2003-1-18	星期六	星期六	休息
01	2003-1-19	星期日	星期日	休息
…	…	…	…	…

⑭ 如表 17-23 所示为质量对应关系

表 17-23 质量对应关系

物料代码	物料名称	数量统计类
10-C001	家用电脑	合格
		返工
		废品
20-C001	主机	合格
		返工
		废品

续表

物料代码	物料名称	数量统计类
30-C001	键盘	合格
		不合格
30-C002	CPU	合格
		不合格
30-C003	硬盘	合格
		不合格
30-C004	内存	合格
		不合格
30-C005	鼠标	合格
		不合格
30-C006	主板	合格
		不合格
30-C007	显示器	合格
		不合格
30-C008	机箱	合格
		不合格

⑮ 如表 17-24 所示为库存。

表 17-24　库存信息

仓库	物料编码	物料名称	库存量	安全库存	可用量	物料批号
成品库	10-C001	家用电脑	60	10	50	
半成品库	20-C001	主机	60	10	40	
原材料库	30-C001	键盘	25	10	8	
	30-C002	CPU	60	10	50	
	30-C003	硬盘	30	10	0	
	30-C004	内存	10	5	5	
	30-C005	鼠标	20	5	15	
	30-C006	主板	50	10	40	
	30-C007	显示器	70	10	60	
	30-C008	机箱	60	10	50	

2．销售管理

针对 ERP 软件流程进行用例设计，对数据的有效性校验可以用等价类划分和边界值分析法进行用例设计，在此不再列举。

（1）流程图

如图 17-4 所示为销售管理流程图。

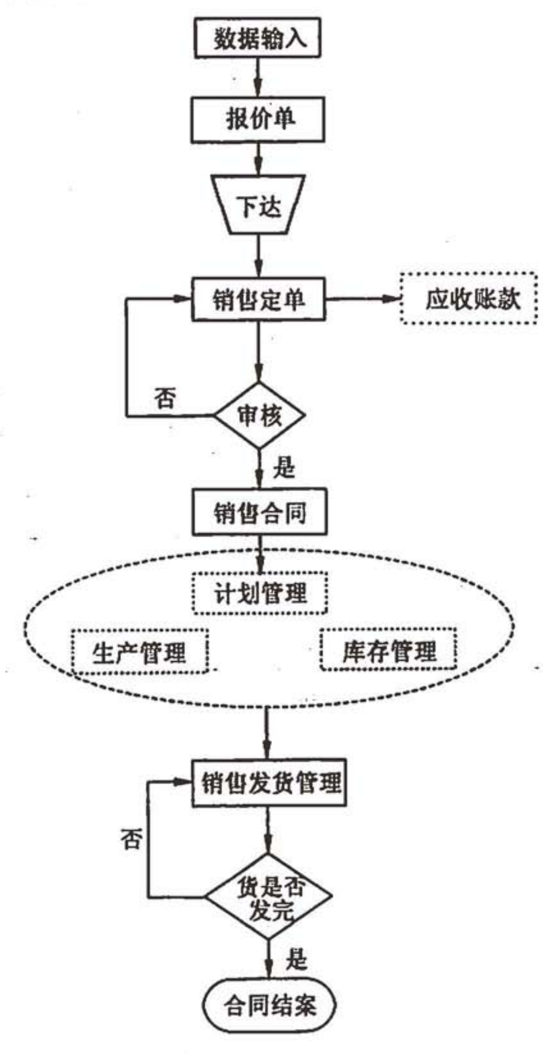


图 17-4 销售管理流程图

（2）用例步骤及数据

如表 17-25 至表 17-30 所示为销售管理模块测试用例的步骤及数据。

表 17-25　生成报价单

序　号	1							
测试项目	生成报价单							
数据来源	手工输入、基础数据							
操作或输入数据	报价单号	客户	总金额	货币	报价日期	发货日期	部门	销售员
	200301001	客户 A	500 000.00	人民币	2003-1-1	2003-1-24	销售一部	销售员甲
	报价单序号	物料号	品名	数量		单位	价格	需求日期
	1	10-C001	家用电脑	100		台	5 000.00	2003-1-23
预期结果	保存成功							
实际结果								

表 17-26　报价单下达

序　号	2
测试项目	报价单下达
数据来源	报价单
操作或输入数据	下达报价单
预期结果	生成在销售定单，销售定单号为 200301001，定单数据与报价单数据一致。除销售定单号外，其他数据项允许修改
实际结果	

表 17-27　销售定单送审 1

序　号	3
测试项目	销售定单送审
数据来源	销售定单
操作或输入数据	将销售定单 200301001 送审
预期结果	送审成功
实际结果	

表 17-28　销售定单审核 1

序　号	4	序　号	4
测试项目	销售定单审核	预期结果	驳回成功
数据来源	销售定单	实际结果	
操作或输入数据	将销售定单 200301001 驳回		



表 17-29 销售定单送审 2

序　　号	5	序　　号	5
测试项目	销售定单送审	预期结果	送审成功
数据来源	销售定单	实际结果	
操作或输入数据	再次将销售定单 200301001 送审		

表 17-30 销售定单审核 2

序　　号	6
测试项目	销售定单审核
数据来源	销售定单
操作或输入数据	销售定单 200301001 审核通过
预期结果	审核成功 生成合同号为 200301001 的销售合同，同时该合同流向计划管理模块（即：在销售需求查看中能够查看到该合同）
实际结果	

以上用例模拟了销售管理模块中制定销售定单最简单的流程，由于数据流向计划管理部门，我们将在下面进行计划管理模块的用例设计，销售管理模块的用例设计到此暂停。

3．计划管理

本部分我们继续场景 10 的用例设计，其中我们会用到因果图的方法进行该模块的测试用例设计。

（1）流程图

计划管理流程图如图 17-5 所示。

设计流程图时，我们不仅考虑了场景 10 的条件组合中涉及到的计划管理条件值（库存中有可用产品但不满足销售需求，且库存有可用零部件但不满足生产需求），也考虑了其他几种情况，这是为了在这里给大家介绍另一种测试用例设计方法——因果图法。

因果图方法的思路是：从用自然语言书写的程序规格说明描述中找出因（输入条件）和果（输出或程序状态的改变），通过因果图转换为判定表。

- 分析原因和结果。

原因和结果分析如图 17-6 所示。

销售管理
计划需求
废品报告
MPS 计算
可用库存是否满足销售需求
是
销售发货管理
否
是否有可用库存
是
销售发货管理
否
主生产计划
MRP 计算
可用库存是否满足生产需求
否
计划请购单
是
计划生产定单
下达
下达
采购管理
生产管理

图 17-5　计划管理流程图

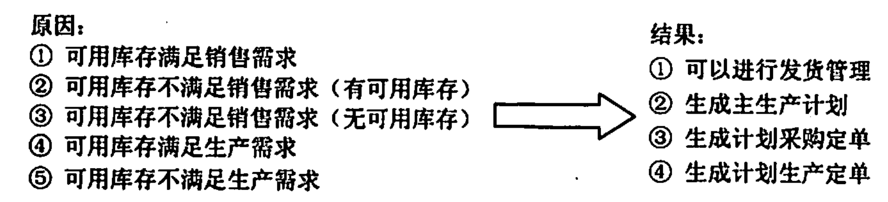


图 17-6　原因与结果分析

- 因果图。

如图 17-7 所示为因果图。

输入条件（原因）

输出条件（结果）

可用库存满足销售需求

可用库存不满足销售需求（有可用库存）

可用库存不满足销售需求（无可用库存）

可用库存满足生产需求

可用库存不满足生产需求

进行发货管理

生成主生产计划

生成计划生产定单

生成计划采购定单

E

I

E

E

O

图 17-7 因果图

- 测试用例

如表 17-31 所示为判定表

如果我们希望对该模块进行更加全面的测试，可以按照表 17-31 进行具体数据的设计。分析表 17-31，我们可以看到 a、b 两组的输出是一样的。这时我们应该分析一下原因，看看在设计用例数据时是不是只考虑其中一个就够了。

结合流程图我们发现实际流程中如果“可用库存满足销售需求”，那么数据直接就转到销售发货管理了，数据根本不会流向下一个判断点，结合等价类划分法，可以得出结论，设计用例数据时只考虑 a、b 两组中的任一个情况即可。

表 17-31　判定表

			a	b	c	d	e	f
输入	可用库存满足销售需求	(1)	1	1	0	0	0	0
	可用库存不满足销售需求（有可用库存）	(2)	0	0	1	1	0	0
	可用库存不满足销售需求（无可用库存）	(3)	0	0	0	0	1	1
	可用库存满足生产需求	(4)	1	0	1	0	1	0
	可用库存不满足生产需求	(5)	0	1	0	1	0	1
输出	可以进行发货管理	(21)	1	1	1	1	0	0
	生成主生产计划	(22)	0	0	1	1	1	1
	生成计划采购定单	(23)	0	0	0	1	0	1
	生成计划生产定单	(24)	0	0	1	1	1	1

然而，为什么会出现这种情况呢？其中因素很多，如软件复杂程度、输入条件的选择、输入条件之间的关联关系等。所以，实际用因果图法进行用例设计时，想要设计出“完美”的“输入”、“输出”是比较困难的。我们应该将各种用例设计方法相结合，融会贯通，才能够设计出好的测试用例。

此处由于篇幅限制，我们只根据情况 d 进行用例数据的设计，情况 d 的输出是最多的，而且和前面选择按第 10 组条件组合进行整个用例的设计也是相符的。

（2）用例步骤及数据

表 17-32 至表 17-38 为计划管理模块测试用例步骤及数据

表 17-32　计划要求

序　号	7
测试项目	计划需求查看
数据来源	销售需求
操作或输入数据	查询计划需求
预期结果	可以查到销售定单 200301001。定单的数据不允许修改
实际结果	

表 17-33　MPS 计算

序　号	8
测试项目	MPS 计算
数据来源	销售定单 200301001、基础数据
操作或输入数据	进行 MPS 计算



续表

<table>
<tr><th>序　号</th><th>8</th></tr>
<tr><td>预期结果</td><td>生成主生产计划（计划开工日期允许修改）：
<table><tr><td>物料代码</td><td>物料名称</td><td>定单数量</td><td>生产数量</td><td>单位</td><td>计划来源</td><td>计划开工日期</td><td>计划完工日期</td><td>计划令号</td><td>排产方式</td></tr><tr><td>10-C001</td><td>家用电脑</td><td>100</td><td>50</td><td>台</td><td>定单</td><td>2003-1-9</td><td>2003-1-23</td><td>20030100001</td><td>倒排</td></tr></table>库存中家用电脑的库存可用量变为 0</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-34　MRP 计算

<table>
<tr><th>序　号</th><th>9</th></tr>
<tr><td>测试项目</td><td>MRP 计算</td></tr>
<tr><td>数据来源</td><td>主生产计划 20030100001、基础数据</td></tr>
<tr><td>操作或输入数据</td><td>进行 MRP 计算</td></tr>
<tr><td>预期结果</td><td>计划生产定单（数据项不允许修改）：
<table><tr><td>物料代码</td><td>物料名称</td><td>生产数量</td><td>单位</td><td>工作中心</td><td>计划开工日期</td><td>计划完工日期</td><td>计划令号</td><td>排产方式</td></tr><tr><td>10-C001</td><td>家用电脑</td><td>50</td><td>台</td><td>总装一线</td><td>2003-1-10</td><td>2003-1-23</td><td>200301000011</td><td>倒排</td></tr></table>
<table><tr><td>物料代码</td><td>物料名称</td><td>生产数量</td><td>单位</td><td>工作中心</td><td>计划开工日期</td><td>计划完工日期</td><td>计划令号</td><td>排产方式</td></tr><tr><td>20-C001</td><td>主机</td><td>10</td><td>台</td><td>加工一线</td><td>2003-1-21</td><td>2003-1-22</td><td>200301000012</td><td>倒排</td></tr></table>计划请购单（数据项不允许修改）：
<table><tr><td>物料代码</td><td>物料名称</td><td>采购数量</td><td>单位</td><td>部门</td><td>计划采购日期</td><td>计划到货日期</td><td>计划令号</td><td>排产方式</td></tr><tr><td>30-C001</td><td>键盘</td><td>50</td><td>个</td><td>采购一部</td><td>2003-1-7</td><td>2003-1-9</td><td>200301000013</td><td>倒排</td></tr></table></td></tr>
</table>

续表

<table>
<tr><th>序　号</th><th>8</th></tr>
<tr><td>预期结果</td><td>
<table>
<tr><th>物料代码</th><th>物料名称</th><th>采购数量</th><th>单位</th><th>部门</th><th>计划采购日期</th><th>计划到货日期</th><th>计划令号</th><th>排产方式</th></tr>
<tr><td>30-C003</td><td>硬盘</td><td>10</td><td>个</td><td>采购一部</td><td>2003-1-16</td><td>2003-1-20</td><td>200301000014</td><td>倒排</td></tr>
</table>
<table>
<tr><th>物料代码</th><th>物料名称</th><th>采购数量</th><th>单位</th><th>部门</th><th>计划采购日期</th><th>计划到货日期</th><th>计划令号</th><th>排产方式</th></tr>
<tr><td>30-C004</td><td>内存</td><td>20</td><td>个</td><td>采购一部</td><td>2003-1-16</td><td>2003-1-20</td><td>200301000015</td><td>倒排</td></tr>
</table>
<table>
<tr><th>物料代码</th><th>物料名称</th><th>采购数量</th><th>单位</th><th>部门</th><th>计划采购日期</th><th>计划到货日期</th><th>计划令号</th><th>排产方式</th></tr>
<tr><td>30-C007</td><td>鼠标</td><td>40</td><td>台</td><td>采购一部</td><td>2003-1-7</td><td>2003-1-9</td><td>200301000016</td><td>倒排</td></tr>
</table>
库存中主机、键盘、硬盘、内存、鼠标的可用量变为 0； CPU 的可用量变为 40；主板的可用量变为 30、显示器的可用量变为 10、机箱的可用量变为 40
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-35　计划生产定单下达

序　号	10
测试项目	下达计划生产定单 200301000011、200301000012
数据来源	计划生产定单
操作或输入数据	下达计划生产定单为车间作业计划
预期结果	下达成功。在生产管理模块的车间作业计划中能够查询到作业计划号为 ZY200301000011 和 ZY200301000012 的作业计划
实际结果	

表 17-36　计划请购单下达

序　号	11
测试项目	下达计划请购单
数据来源	计划请购单 200301000013、200301000014、200301000015、200301000016
操作或输入数据	下达计划生产定单为计划采购定单
预期结果	下达成功。在采购管理模块的计划采购定单中能够查询到定单号为 CG200301000013、CG200301000014 和 CG200301000015、CG200301000016 的计划采购定单
实际结果	



说明：这部分的测试用例模拟了计划管理模块根据计划需求、物料库存量及其他基础数据按照一定的算法进行计算，得到计划生产定单、计划请购单并进行下达的过程。其中 MPS 计算、MRP 计算的算法较复杂，而且各 ERP 软件也不同，需要参考软件设计说明书。

从期初库存可以看出，库存中有 50 台家用电脑可以用，在进行完 MPS 计算后这 50 台电脑已经分配给了销售定单 200301001，这时在销售管理模块中已经可以对这 50 台电脑进行发货了。

表 17-37 销售发货单 1

序 号	12
测试项目	销售发货单
数据来源	销售定单、基础数据、计划管理
操作或输入数据	生成销售发货单
预期结果	销售发货单： （见下表）
实际结果	

发货单号	定单号	仓库	名称	定单数量	已发数量	此次发货数量	发货日期	销售员
FG200301001	200301000004	成品库	家用电脑	100	0	50	2003-1-7	销售员甲

表 17-38 销售出库 1

序 号	13
测试项目	销售出库
数据来源	销售发货单
操作或输入数据	根据销售发货单 FG200301001 进行销售出库操作
预期结果	查询库存中物料的数量为： （见下表）
实际结果	

仓库	物料编码	物料名称	库存量	安全库存	可用量	物料批号
成品库	10-C001	家用电脑	10	10	0	

4．采购管理

（1）流程图

采购管理流程图如图 17-8 所示。

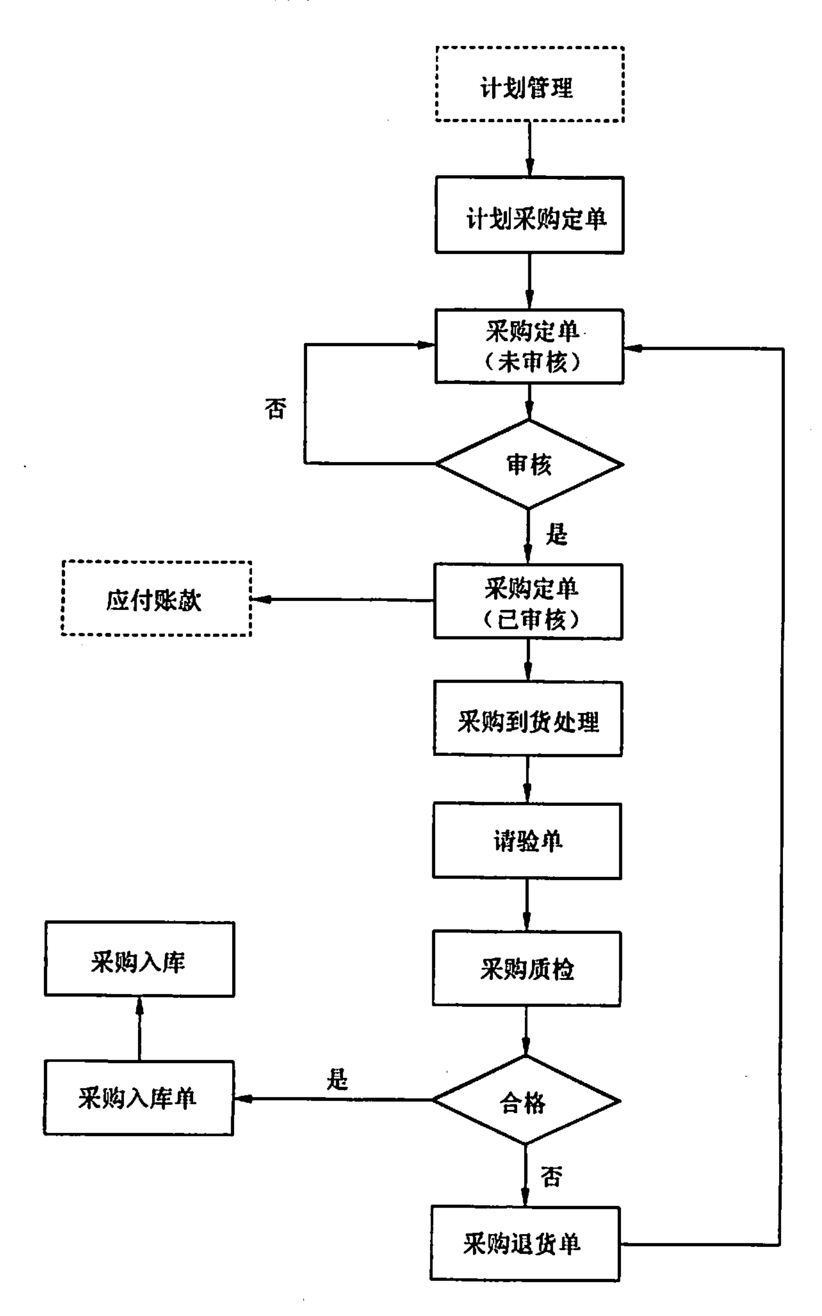


图 17-8　采购管理流程图

（2）用例步骤及数据

如表 17-39 至表 17-48 所示为采购管理模块测试用例步骤及数据。

表 17-39 计划采购定单

<table>
<tr><th>序 号</th><th>14</th></tr>
<tr><td>测试项目</td><td>计划采购定单查询</td></tr>
<tr><td>数据来源</td><td>计划请购单、基础数据</td></tr>
<tr><td>操作或输入数据</td><td>查询计划采购定单</td></tr>
<tr><td>预期结果</td><td>查询到计划采购定单：
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>采购日期</th><th>到货日期</th></tr>
<tr><td>CG200301000013</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>100.00</td><td>2003-1-7</td><td>2003-1-9</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>采购日期</th><th>到货日期</th></tr>
<tr><td>CG200301000014</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td><td>1000.00</td><td>2003-1-16</td><td>2003-1-20</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>采购日期</th><th>到货日期</th></tr>
<tr><td>CG200301000015</td><td>内存</td><td>原材料</td><td>外购</td><td>20 个</td><td>100.00</td><td>2003-1-16</td><td>2003-1-20</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>采购日期</th><th>到货日期</th></tr>
<tr><td>CG200301000016</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td><td>50</td><td>2003-1-7</td><td>2003-1-9</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-40 计划采购定单确认

<table>
<tr><th>序 号</th><th>15</th></tr>
<tr><td>测试项目</td><td>计划采购定单确认</td></tr>
<tr><td>数据来源</td><td>计划采购定单</td></tr>
<tr><td>操作或输入数据</td><td>确认计划采购定单 CG200301000013、CG200301000014、CG200301000015、CG200301000016 生成采购定单</td></tr>
<tr><td>预期结果</td><td>生成采购定单（数据项供应商、单价和数量可以修改）：
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>收货日期</th><th>状态</th></tr>
<tr><td>CG200301000013</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>200</td><td>10000</td><td>供应商 B</td><td>2003-1-7</td><td>2003-1-9</td><td>未审核</td></tr>
</table>
</td></tr>
</table>

续表

<table>
<tr><th>序　号</th><th>15</th></tr>
<tr><td>预期结果</td><td>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>到货日期</th><th>状态</th></tr>
<tr><td>CG200301000014</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td><td>1000</td><td>10000</td><td>供应商 B</td><td>2003-1-16</td><td>2003-1-20</td><td>未审核</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>到货日期</th><th>状态</th></tr>
<tr><td>CG200301000015</td><td>内存</td><td>原材料</td><td>外购</td><td>20 个</td><td>100</td><td>2000</td><td>供应商 B</td><td>2003-1-16</td><td>2003-1-20</td><td>未审核</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>到货日期</th><th>状态</th></tr>
<tr><td>CG200301000016</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td><td>50</td><td>2000</td><td>供应商 A</td><td>2003-1-7</td><td>2003-1-9</td><td>未审核</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-41　采购定单审核 1

<table>
<tr><th>序　号</th><th>16</th></tr>
<tr><td>测试项目</td><td>采购定单审核</td></tr>
<tr><td>数据来源</td><td>采购定单</td></tr>
<tr><td>操作或输入数据</td><td>对采购定单 CG200301000014、CG200301000015、CG200301000016 进行审核操作；对采购定单 CG200301000013 进行驳回操作</td></tr>
<tr><td>预期结果</td><td>状态为“已审核”的采购定单不可以再修改；状态为“驳回”的采购定单在上一级修改后，可再送审
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>收货日期</th><th>状态</th></tr>
<tr><td>CG200301000013</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>200</td><td>10000</td><td>供应商 B</td><td>2003-1-7</td><td>2003-1-9</td><td>驳回</td></tr>
</table>
<table>
<tr><th>定单号</th><th>名称</th><th>类型</th><th>来源</th><th>数量</th><th>单价（元）</th><th>总额</th><th>供应商</th><th>采购日期</th><th>到货日期</th><th>状态</th></tr>
<tr><td>CG200301000014</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td><td>1000</td><td>10000</td><td>供应商 B</td><td>2003-1-16</td><td>2003-1-20</td><td>已审核</td></tr>
</table>
</td></tr>
</table>



续表

<table>
<tr><td>序　号</td><td>16</td></tr>
<tr><td>预期结果</td><td>
<table>
<tr><td>定单号</td><td>名称</td><td>类型</td><td>来源</td><td>数量</td><td>单价（元）</td><td>总额</td><td>供应商</td><td>采购日期</td><td>到货日期</td><td>状态</td></tr>
<tr><td>CG200301000015</td><td>内存</td><td>原材料</td><td>外购</td><td>20 个</td><td>100</td><td>2000</td><td>供应商B</td><td>2003-1-16</td><td>2003-1-20</td><td>已审核</td></tr>
</table>
<table>
<tr><td>定单号</td><td>名称</td><td>类型</td><td>来源</td><td>数量</td><td>单价（元）</td><td>总额</td><td>供应商</td><td>采购日期</td><td>到货日期</td><td>状态</td></tr>
<tr><td>CG200301000016</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td><td>50</td><td>2000</td><td>供应商 A</td><td>2003-1-7</td><td>2003-1-9</td><td>已审核</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-42　采购定单审核 2

<table>
<tr><td>序　号</td><td>17</td></tr>
<tr><td>测试项目</td><td>采购定单审核</td></tr>
<tr><td>数据来源</td><td>采购定单</td></tr>
<tr><td>操作或输入数据</td><td>对修改后的采购定单 CG200301000013 进行审核操作</td></tr>
<tr><td>预期结果</td><td>状态为“已审核”的采购定单不可以再修改；状态为“驳回”的采购定单在上一级修改后，可再送审
<table>
<tr><td>定单号</td><td>名称</td><td>类型</td><td>来源</td><td>数量</td><td>单价（元）</td><td>总额</td><td>供应商</td><td>采购日期</td><td>收货日期</td><td>状态</td></tr>
<tr><td>CG200301000013</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>100</td><td>5000</td><td>供应商 A</td><td>2003-1-7</td><td>2003-1-9</td><td>已审核</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-43　采购收货单

序　号	18
测试项目	采购收货单
数据来源	采购定单、手工录入
操作或输入数据	根据采购定单 CG200301000014、CG200301000015、CG200301000016、CG200301000013 生成收货单（此处暂不考虑日期）

续表

序　号	18
预期结果	生成收货单：

相关采购定单号	收货单号	名称	类型	来源	应收数量	实收数量	供应商	仓库	收获日期
CG200301000013	SH000001	键盘	原材料	外购	50 个	50 个	供应商A	原材料库	2003-1-9

相关采购定单号	收货单号	名称	类型	来源	应收数量	实收数量	供应商	仓库	收货日期
CG200301000016	SH000002	鼠标	原材料	外购	40 个	40 个	供应商 A	原材料库	2003-1-9

相关采购定单号	收货单号	名称	类型	来源	应收数量	实收数量	供应商	仓库	收货日期
CG200301000014	SH000003	硬盘	原材料	外购	10 个	10 个	供应商 B	原材料库	2003-1-20

相关采购定单号	收货单号	名称	类型	来源	应收数量	实收数量	供应商	仓库	收货日期
CG200301000015	SH000004	内存	原材料	外购	20 个	20 个	供应商B	原材料库	2003-1-20

实际结果	

表 17-44　采购请验单

序　号	19
测试项目	采购请验单
数据来源	采购收货单
操作或输入数据	根据采购收货单 SH000001、SH000002、SH000003、SH00004 生成请验单
预期结果	生成请验单：

相关收货单号	请验单号	名称	类型	来源	送检数量
SH000001	QY000001	键盘	原材料	外购	50 个



续表

<table>
<tr><th>序　号</th><th>19</th></tr>
<tr><td>预期结果</td><td>
<table><tr><td>相关收货单号</td><td>请验单号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td></tr><tr><td>SH000002</td><td>QY000002</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td></tr></table>
<table><tr><td>相关收货单号</td><td>请验单号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td></tr><tr><td>SH000003</td><td>QY000003</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td></tr></table>
<table><tr><td>相关收货单号</td><td>请验单号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td></tr><tr><td>SH000004</td><td>QY000004</td><td>内存</td><td>原材料</td><td>外购</td><td>20 个</td></tr></table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-45　采购检验报告

<table>
<tr><th>序　号</th><th>20</th></tr>
<tr><td>测试项目</td><td>采购检验报告</td></tr>
<tr><td>数据来源</td><td>请验单；手工录入</td></tr>
<tr><td>操作或输入数据</td><td>生成采购检验报告，内存不合格数量为 3，其他全部合格</td></tr>
<tr><td>预期结果</td><td>生成采购检验报告：
<table><tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr><tr><td>QY000001</td><td>BG000001</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>50 个</td><td>0</td><td></td></tr></table>
<table><tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr><tr><td>QY000002</td><td>BG000002</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td><td>40 个</td><td>0</td><td></td></tr></table>
<table><tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr><tr><td>QY000003</td><td>BG000003</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td><td>10 个</td><td>0</td><td></td></tr></table>
<table><tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr><tr><td>QY000004</td><td>BG000004</td><td>内存</td><td>原材料</td><td>外购</td><td>20 个</td><td>17 个</td><td>3</td><td>不能正常工作</td></tr></table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-46　采购入库单

<table>
<tr><td>序　号</td><td>21</td></tr>
<tr><td>测试项目</td><td>采购入库单</td></tr>
<tr><td>数据来源</td><td>采购检验报告</td></tr>
<tr><td>操作或输入数据</td><td>根据检验报告 BG000001、BG000002、BG000003、BG000004 生成采购入库单（数据项不允许修改）</td></tr>
<tr><td>预期结果</td><td>生成采购入库单：
<table>
<tr><td>报告号</td><td>入库单号</td><td>名称</td><td>类型</td><td>来源</td><td>入库数量</td></tr>
<tr><td>BG000001</td><td>RK000001</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td></tr>
<tr><td>报告号</td><td>入库单号</td><td>名称</td><td>类型</td><td>来源</td><td>入库数量</td></tr>
<tr><td>BG000002</td><td>RK000002</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td></tr>
<tr><td>报告号</td><td>入库单号</td><td>名称</td><td>类型</td><td>来源</td><td>入库数量</td></tr>
<tr><td>BG000003</td><td>RK000003</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td></tr>
<tr><td>报告号</td><td>入库单号</td><td>名称</td><td>类型</td><td>来源</td><td>入库数量</td></tr>
<tr><td>BG000004</td><td>RK000004</td><td>内存</td><td>原材料</td><td>外购</td><td>17 个</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-47　采购入库

<table>
<tr><td>序　号</td><td>22</td></tr>
<tr><td>测试项目</td><td>采购入库</td></tr>
<tr><td>数据来源</td><td>采购入库单</td></tr>
<tr><td>操作或输入数据</td><td>进行采购入库操作</td></tr>
<tr><td>预期结果</td><td>查询库存中物料的数量为：
<table>
<tr><td>仓库</td><td>物料编码</td><td>物料名称</td><td>库存量</td><td>安全库存</td><td>可用量</td><td>物料批号</td></tr>
<tr><td rowspan="8">原材料库</td><td>30-C001</td><td>键盘</td><td>75</td><td>10</td><td>8</td><td></td></tr>
<tr><td>30-C002</td><td>CPU</td><td>60</td><td>10</td><td>40</td><td></td></tr>
<tr><td>30-C003</td><td>硬盘</td><td>40</td><td>10</td><td>0</td><td></td></tr>
<tr><td>30-C004</td><td>内存</td><td>27</td><td>5</td><td>2</td><td></td></tr>
<tr><td>30-C005</td><td>鼠标</td><td>60</td><td>5</td><td>5</td><td></td></tr>
<tr><td>30-C006</td><td>主板</td><td>50</td><td>10</td><td>30</td><td></td></tr>
<tr><td>30-C007</td><td>显示器</td><td>70</td><td>10</td><td>10</td><td></td></tr>
<tr><td>30-C008</td><td>机箱</td><td>60</td><td>10</td><td>40</td><td></td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

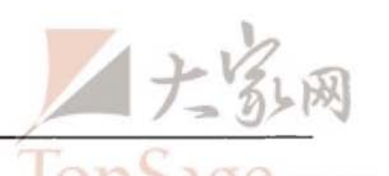


表 17-48 采购退货单

序　号	23
测试项目	采购退货单
数据来源	采购检验报告
操作或输入数据	根据采购检验报告 BG000004 生成采购退货单
预期结果	采购退货单： 检验报告号：BG000004；退货单号：TH000004；名称：内存；类型：原材料；来源：外购；退货数量：3 个；退货原因：质检不合格
实际结果	

预期结果中的采购退货单：

检验报告号	退货单号	名称	类型	来源	退货数量	退货原因
BG000004	TH000004	内存	原材料	外购	3 个	质检不合格

说明：这部分测试用例模拟了，采购管理模块根据计划管理模块下达的计划采购单，进行物料采购、采购质检和采购入库的过程。

在进行采购收货单测试时，我们为了用例设计方便，在不影响测试效果的前提下，暂不考虑时间问题。在质检报告中有 3 个不合格的内存，对这 3 个内存，生成退货单退货后，可以根据退货单，再生成采购定单，此处不再举例。

5. 生产管理

（1）流程图

如图 17-9 所示为生产管理流程图。

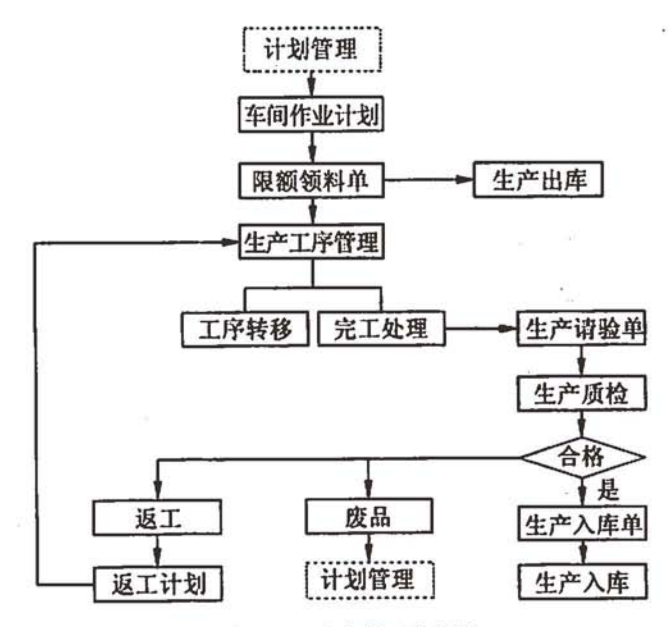


图 17-9 生产管理流程图

（2）用例步骤及数据

如表 17-49 至表 17-68 所示为生产管理模块测试用例及数据。

表 17-49　车间作业计划

<table>
<tr><td>序　号</td><td>24</td></tr>
<tr><td>测试项目</td><td>车间作业计划查询</td></tr>
<tr><td>数据来源</td><td>计划生产定单</td></tr>
<tr><td>操作或输入数据</td><td>查询车间作业计划</td></tr>
<tr><td>预期结果</td><td>查询到 ZY200301000011：
<table>
<tr><td>物料代码</td><td>物料名称</td><td>生产数量</td><td>单位</td><td>工作中心</td><td>工序名称</td><td>计划开工日期</td><td>计划完工日期</td><td>作业计划号</td></tr>
<tr><td>10-C001</td><td>家用电脑</td><td>50</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-10</td><td>2003-1-23</td><td>ZY200301000011</td></tr>
</table>
查询到 ZY200301000012：
<table>
<tr><td>物料代码</td><td>物料名称</td><td>生产数量</td><td>单位</td><td>工作中心</td><td>工序名称</td><td>计划开工日期</td><td>计划完工日期</td><td>作业计划号</td></tr>
<tr><td>20-C001</td><td>主机</td><td>10</td><td>台</td><td>加工一线</td><td>加工</td><td>2003-1-21</td><td>2003-1-22</td><td>ZY200301000012</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-50　限额领料单

<table>
<tr><td>序　号</td><td>25</td></tr>
<tr><td>测试项目</td><td>限额领料单</td></tr>
<tr><td>数据来源</td><td>车间作业计划、基础数据</td></tr>
<tr><td>操作或输入数据</td><td>生成限额领料单</td></tr>
<tr><td>预期结果</td><td>生成限额领料单：
<table>
<tr><td>领料单号</td><td>作业计划号</td><td>物料编码</td><td>名称</td><td>应发数量</td><td>已领数量</td><td>领料部门</td></tr>
<tr><td rowspan="4">LL000011</td><td rowspan="4">ZY200301000011</td><td>20-C001</td><td>主机</td><td>40</td><td>0</td><td rowspan="4">总装一线</td></tr>
<tr><td>30-C001</td><td>键盘</td><td>50</td><td>0</td></tr>
<tr><td>30-C005</td><td>鼠标</td><td>50</td><td>0</td></tr>
<tr><td>30-C007</td><td>显示器</td><td>50</td><td>0</td></tr>
</table>
</td></tr>
</table>



续表

序号	25

预期结果：

领料单号	作业计划号	物料编码	名称	应发数量	已领数量	领料部门
LL000012	ZY200301000012	20-C002	CPU	10	0	加工一线
		30-C003	硬盘	10	0	
		30-C004	内存	20	0	
		30-C006	主板	10	0	
		30-C008	机箱	10	0	

实际结果	

说明：限额领料单中主机的应发数量是 40 台，比实际需要少了 10 台，这是因为另外 10 台是由上一道工序（即：加工一线）生产、通过工序转移交给总装一线的。

表 17-51 生产出库

序号	26
测试项目	生产出库
数据来源	限额领料单、库存数据
操作或输入数据	进行生产出库操作

预期结果：

查询限额领料单：

领料单号	作业计划号	物料编码	名称	应发数量	已领数量	领料部门
LL000011	ZY200301000011	20-C001	主机	40	40	总装一线
		30-C001	键盘	50	50	
		30-C005	鼠标	50	50	
		30-C007	显示器	50	50	

领料单号	作业计划号	物料编码	名称	应发数量	已领数量	领料部门
LL000012	ZY200301000012	20-C002	CPU	10	10	加工一线
		30-C003	硬盘	10	10	
		30-C004	内存	20	20	
		30-C006	主板	10	10	
		30-C008	机箱	10	10	

续表

<table>
<tr><th>序　号</th><th>26</th></tr>
<tr><td>预期结果</td><td>查询库存：
<table>
<tr><th>仓库</th><th>物料编码</th><th>物料名称</th><th>库存量</th><th>安全库存</th><th>可用量</th><th>物料批号</th></tr>
<tr><td>半成品库</td><td>20-C001</td><td>主机</td><td>20</td><td>10</td><td>0</td><td></td></tr>
<tr><td rowspan="8">原材料库</td><td>30-C001</td><td>键盘</td><td>25</td><td>10</td><td>8</td><td></td></tr>
<tr><td>30-C002</td><td>CPU</td><td>50</td><td>10</td><td>40</td><td></td></tr>
<tr><td>30-C003</td><td>硬盘</td><td>30</td><td>10</td><td>0</td><td></td></tr>
<tr><td>30-C004</td><td>内存</td><td>7</td><td>5</td><td>2</td><td></td></tr>
<tr><td>30-C005</td><td>鼠标</td><td>10</td><td>5</td><td>5</td><td></td></tr>
<tr><td>30-C006</td><td>主板</td><td>40</td><td>10</td><td>30</td><td></td></tr>
<tr><td>30-C007</td><td>显示器</td><td>20</td><td>10</td><td>10</td><td></td></tr>
<tr><td>30-C008</td><td>机箱</td><td>50</td><td>10</td><td>40</td><td></td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-52　工序报告单 1

<table>
<tr><th>序　号</th><th>27</th></tr>
<tr><td>测试项目</td><td>工序报告单</td></tr>
<tr><td>数据来源</td><td>车间作业计划 ZY200301000011</td></tr>
<tr><td>操作或输入数据</td><td>生成工序报告单，为了简化过程，已产数量直接设为 40</td></tr>
<tr><td>预期结果</td><td>工序报告单：
<table>
<tr><th>报告单号</th><th>物料名称</th><th>生产数量</th><th>已产数量</th><th>单位</th><th>工作中心</th><th>工序名称</th><th>计划开工日期</th><th>计划完工日期</th><th>实际开工时间</th><th>实际完工时间</th><th>完工</th></tr>
<tr><td>GX200301000011</td><td>家用电脑</td><td>50</td><td>40</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-10</td><td>2003-1-23</td><td>2003-1-10</td><td></td><td>否</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>



表 17-53 工序报告单 2

序 号	28
测试项目	工序报告单
数据来源	车间作业计划 ZY200301000012
操作或输入数据	生成工序报告单，已产数量填 5
预期结果	工序报告单：（见下表）
实际结果	

报告单号	物料名称	生产数量	已产数量	单位	工作中心	工序名称	计划开工日期	计划完工日期	实际开工时间	实际完工时间	完工
GX200301000012	主机	10	5	台	总装一线	总装	2003-1-21	2003-1-22	2003-1-21		否

表 17-54 工序转移单 1

序 号	29
测试项目	工序转移单
数据来源	工序报告单
操作或输入数据	生成工序转移单
预期结果	工序转移单：（见下表）
实际结果	

当前工序	下一工序	物料	已生产数量	可转移数量	转移数量	是否质检	转移日期
加工	总装	主机	5	5	5	否	2003-1-22

表 17-55 工序接收单 1

序 号	30
测试项目	工序接收单
数据来源	工序转移单
操作或输入数据	生成工序接收单
预期结果	工序接收单：（见下表）
实际结果	

当前工序	上一工序	物料	接收数量	是否质检	接收时间
总装	加工	主机	5	否	2003-1-22

表 17-56　工序报告单 3

<table>
<tr><th>序　号</th><th>31</th></tr>
<tr><td>测试项目</td><td>工序报告单</td></tr>
<tr><td>数据来源</td><td>车间作业计划 ZY200301000011</td></tr>
<tr><td>操作或输入数据</td><td>修改工序报告单中的已产数量为 45</td></tr>
<tr><td>预期结果</td><td>工序报告单：
<table>
<tr><th>报告单号</th><th>物料名称</th><th>生产数量</th><th>已产数量</th><th>单位</th><th>工作中心</th><th>工序名称</th><th>计划开工日期</th><th>计划完工日期</th><th>实际开工时间</th><th>实际完工时间</th><th>完工</th></tr>
<tr><td>GX200301000011</td><td>家用电脑</td><td>50</td><td>45</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-10</td><td>2003-1-23</td><td>2003-1-10</td><td></td><td>否</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-57　工序报告单 4

<table>
<tr><th>序　号</th><th>32</th></tr>
<tr><td>测试项目</td><td>工序报告单</td></tr>
<tr><td>数据来源</td><td>车间作业计划 ZY200301000012</td></tr>
<tr><td>操作或输入数据</td><td>修改工序报告单已产数量为 10
置完工</td></tr>
<tr><td>预期结果</td><td>工序报告单：
<table>
<tr><th>报告单号</th><th>物料名称</th><th>生产数量</th><th>已产数量</th><th>单位</th><th>工作中心</th><th>工序名称</th><th>计划开工日期</th><th>计划完工日期</th><th>实际开工时间</th><th>实际完工时间</th><th>完工</th></tr>
<tr><td>GX200301000012</td><td>主机</td><td>10</td><td>10</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-21</td><td>2003-1-22</td><td>2003-1-21</td><td>2003-1-22</td><td>是</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-58　工序转移单 2

序　号	33
测试项目	工序转移单
数据来源	工序报告单
操作或输入数据	生成工序转移单



续表

<table>
<tr><td>序　号</td><td colspan="8">33</td></tr>
<tr><td rowspan="3">预期结果</td><td colspan="8">工序转移单：</td></tr>
<tr><td>当前工序</td><td>下一工序</td><td>物料</td><td>已生产数量</td><td>可转移数量</td><td>转移数量</td><td>是否质检</td><td>转移日期</td></tr>
<tr><td>加工</td><td>总装</td><td>主机</td><td>10</td><td>5</td><td>5</td><td>否</td><td>2003-1-23</td></tr>
<tr><td>实际结果</td><td colspan="8"></td></tr>
</table>

表 17-59　工序接收单 2

<table>
<tr><td>序　号</td><td colspan="6">34</td></tr>
<tr><td>测试项目</td><td colspan="6">工序接收单</td></tr>
<tr><td>数据来源</td><td colspan="6">工序转移单：</td></tr>
<tr><td>操作或输入数据</td><td colspan="6">生成工序接收单</td></tr>
<tr><td rowspan="3">预期结果</td><td colspan="6">工序接收单：</td></tr>
<tr><td>当前工序</td><td>上一工序</td><td>物料</td><td>接收数量</td><td>是否质检</td><td>接收时间</td></tr>
<tr><td>总装</td><td>加工</td><td>主机</td><td>5</td><td>否</td><td>2003-1-23</td></tr>
<tr><td>实际结果</td><td colspan="6"></td></tr>
</table>

表 17-60　工序报告单 5

<table>
<tr><td>序　号</td><td colspan="13">35</td></tr>
<tr><td>测试项目</td><td colspan="13">工序报告单</td></tr>
<tr><td>数据来源</td><td colspan="13">车间作业计划 ZY200301000011</td></tr>
<tr><td>操作或输入数据</td><td colspan="13">修改工序报告单中的已产数量为 50
置完工</td></tr>
<tr><td rowspan="3">预期结果</td><td colspan="13">工序报告单：</td></tr>
<tr><td>报告单号</td><td>物料名称</td><td>生产数量</td><td>已产数量</td><td>单位</td><td>工作中心</td><td>工序名称</td><td>计划开工日期</td><td>计划完工日期</td><td>实际开工时间</td><td>实际完工时间</td><td>完工</td></tr>
<tr><td>GX200301000011</td><td>家用电脑</td><td>50</td><td>50</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-10</td><td>2003-1-23</td><td>2003-1-10</td><td>2003-1-23</td><td>是</td></tr>
<tr><td>实际结果</td><td colspan="13"></td></tr>
</table>

表 17-61　请验单

<table>
<tr><td>序　号</td><td>36</td></tr>
<tr><td>测试项目</td><td>请验单</td></tr>
<tr><td>数据来源</td><td>工序报告单</td></tr>
<tr><td>操作或输入数据</td><td>生成请验单</td></tr>
<tr><td>预期结果</td><td>请验单：
<table>
<tr><td>相关报告单号</td><td>生产请验单号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td></tr>
<tr><td>GX200301000011</td><td>QY000005</td><td>家用电脑</td><td>原材料</td><td>生产</td><td>50 个</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

表 17-62　检验报告

<table>
<tr><td>序　号</td><td>37</td></tr>
<tr><td>测试项目</td><td>检验报告</td></tr>
<tr><td>数据来源</td><td>请验单、手工录入</td></tr>
<tr><td>操作或输入数据</td><td>生成检验报告，不合格数量为 5，其中返工 2，废品 3</td></tr>
<tr><td>预期结果</td><td>检验报告：
<table>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td></tr>
<tr><td>QY000001</td><td>BG000005</td><td>家用电脑</td><td>产成品</td><td>制造</td><td>50 台</td><td>45 台</td><td>5 台</td></tr>
</table>
废品报告
<table>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>数量</td><td>原因</td></tr>
<tr><td>QY000001</td><td>BG0000051</td><td>家用电脑</td><td>产成品</td><td>制造</td><td>3 台</td><td>无法正常工作</td></tr>
</table>
返工报告
<table>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>数量</td><td>原因</td></tr>
<tr><td>QY000001</td><td>BG0000052</td><td>家用电脑</td><td>产成品</td><td>制造</td><td>2 台</td><td>螺丝没有上紧</td></tr>
</table></td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>



表 17-63 生产入库单

<table>
<tr><td>序 号</td><td colspan="6">38</td></tr>
<tr><td>测试项目</td><td colspan="6">生产入库单</td></tr>
<tr><td>数据来源</td><td colspan="6">生产检验报告</td></tr>
<tr><td>操作或输入数据</td><td colspan="6">生成生产入库单（数据项不允许修改）</td></tr>
<tr><td rowspan="3">预期结果</td><td colspan="6">生产入库单：</td></tr>
<tr><td>报告号</td><td>入库单号</td><td>名称</td><td>类型</td><td>来源</td><td>入库数量</td></tr>
<tr><td>BG000005</td><td>RK000005</td><td>家用电脑</td><td>产成品</td><td>制造</td><td>45 个</td></tr>
<tr><td>实际结果</td><td colspan="6"></td></tr>
</table>

表 17-64 生产入库

<table>
<tr><td>序 号</td><td colspan="7">39</td></tr>
<tr><td>测试项目</td><td colspan="7">生产入库</td></tr>
<tr><td>数据来源</td><td colspan="7">生产入库单 RK000005</td></tr>
<tr><td>操作或输入数据</td><td colspan="7">进行生产入库操作</td></tr>
<tr><td rowspan="3">预期结果</td><td colspan="7">查询库存中物料的数量为：</td></tr>
<tr><td>仓库</td><td>物料编码</td><td>物料名称</td><td>库存量</td><td>安全库存</td><td>可用量</td><td>物料批号</td></tr>
<tr><td>成品库</td><td>10-C001</td><td>家用电脑</td><td>55</td><td>10</td><td>45</td><td></td></tr>
<tr><td>实际结果</td><td colspan="7"></td></tr>
</table>

表 17-65 废品报告

序 号	40
测试项目	废品报告正确流转
数据来源	废品报告
操作或输入数据	查询计划需求
预期结果	可以查到废品报告 BG0000051，数据项不允许修改
实际结果	

表 17-66　返工计划

<table>
<tr><td>序　号</td><td>41</td></tr>
<tr><td>测试项目</td><td>返工计划</td></tr>
<tr><td>数据来源</td><td>返工报告</td></tr>
<tr><td>操作或输入数据</td><td>根据返工报告 BG0000052 生成返工计划</td></tr>
<tr><td>预期结果</td><td>返工计划：
<table>
<tr><th>物料代码</th><th>物料名称</th><th>返工数量</th><th>单位</th><th>工作中心</th><th>工序名称</th><th>计划开工日期</th><th>计划完工日期</th><th>返工计划号</th></tr>
<tr><td>10-C001</td><td>家用电脑</td><td>2</td><td>台</td><td>总装一线</td><td>总装</td><td>2003-1-25</td><td>2003-1-26</td><td>FG200301000011</td></tr>
</table>
<table>
<tr><th>物料代码</th><th>物料名称</th><th>返工数量</th><th>单位</th><th>工作中心</th><th>工序名称</th><th>计划开工日期</th><th>计划完工日期</th><th>返工计划号</th></tr>
<tr><td>20-C001</td><td>主机</td><td>2</td><td>台</td><td>加工一线</td><td>加工</td><td>2003-1-24</td><td>2003-1-25</td><td>FG200301000012</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>

说明：这部分测试用例模拟了，生产管理模块根据计划管理模块下达的车间作业计划，进行生产领料、工序转移、生产完工、生产质检、生产入库的过程。

现在我们应该转入销售发货管理模块，由于生产中出现 5 台不合格产品，所以为了不影响发货日期，使用安全库存，5 台不合格产品重新生产以后，再将安全库存补上。

表 17-67　销售发货单 2

<table>
<tr><td>序　号</td><td>42</td></tr>
<tr><td>测试项目</td><td>销售发货单</td></tr>
<tr><td>数据来源</td><td>生产入库单、销售定单、基础数据</td></tr>
<tr><td>操作或输入数据</td><td>根据销售定单 200301001、生产入库单 RK000005 生成销售发货单</td></tr>
<tr><td>预期结果</td><td>销售发货单：
<table>
<tr><th>发货单号</th><th>定单号</th><th>仓库</th><th>名称</th><th>定单数量</th><th>已发数量</th><th>此次发货数量</th><th>发货日期</th><th>销售员</th></tr>
<tr><td>FG200301001</td><td>200301000004</td><td>成品库</td><td>家用电脑</td><td>100</td><td>50</td><td>50</td><td>2003-01-24</td><td>销售员甲</td></tr>
</table>
</td></tr>
<tr><td>实际结果</td><td></td></tr>
</table>



表 17-68 销售出库 2

<table>
<tr><td>序 号</td><td colspan="7">43</td></tr>
<tr><td>测试项目</td><td colspan="7">销售出库</td></tr>
<tr><td>数据来源</td><td colspan="7">销售发货单</td></tr>
<tr><td>操作或输入数据</td><td colspan="7">根据销售发货单 FG200301001 进行销售出库操作</td></tr>
<tr><td rowspan="4">预期结果</td><td colspan="7">查询库存中物料的数量为：</td></tr>
<tr><td>仓库</td><td>物料编码</td><td>物料名称</td><td>库存量</td><td>安全库存</td><td>可用量</td><td>物料批号</td></tr>
<tr><td>成品库</td><td>10-C001</td><td>家用电脑</td><td>5</td><td>5</td><td>0</td><td></td></tr>
<tr><td colspan="7">合同自动结案</td></tr>
<tr><td>实际结果</td><td colspan="7"></td></tr>
</table>

在实际测试中还会经常用到错误推测法，错误推测设计方法就是基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性地设计测试用例的方法。这种方法对测试工程师的要求较高，需要用丰富的测试经验以及行业知识。例如：在测试采购检验报告时，就可以推测该功能在逻辑性校验方面可能出错，大部分软件在数据类型方面都作了校验，但逻辑性校验做得不太好。如表 17-69 所示为逻辑性检验测试案例。

表 17-69 逻辑性检验测试案例

<table>
<tr><td>序 号</td><td colspan="9">1</td></tr>
<tr><td>测试项目</td><td colspan="9">采购检验报告</td></tr>
<tr><td>数据来源</td><td colspan="9">请验单；手工录入</td></tr>
<tr><td>操作或输入数据</td><td colspan="9">生成采购检验报告，内存合格数量为 18，不合格数量为 5，其他全部合格</td></tr>
<tr><td rowspan="8">预期结果</td><td colspan="9">生成采购检验报告：</td></tr>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr>
<tr><td>QY000001</td><td>BG000001</td><td>键盘</td><td>原材料</td><td>外购</td><td>50 个</td><td>50 个</td><td>0</td><td></td></tr>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr>
<tr><td>QY000002</td><td>BG000002</td><td>鼠标</td><td>原材料</td><td>外购</td><td>40 个</td><td>40 个</td><td>0</td><td></td></tr>
<tr><td>请验单号</td><td>报告号</td><td>名称</td><td>类型</td><td>来源</td><td>送检数量</td><td>合格数量</td><td>不合格数量</td><td>原因</td></tr>
<tr><td>QY000003</td><td>BG000003</td><td>硬盘</td><td>原材料</td><td>外购</td><td>10 个</td><td>10 个</td><td>0</td><td></td></tr>
<tr><td colspan="9">保存内存的检验报告时系统提示：“总数大于送检数量”。保存不成功</td></tr>
<tr><td>实际结果</td><td colspan="9"></td></tr>
</table>

这个例子是用错误推测法设计功能点的测试案例。在对业务逻辑进行测试时我们也可以用到错误推测法。例如：有一销售定单需求计算机数量是 100 台交货日期为 2003 年 3 月 12 日，库存可用量为零，有由预测计划产生的生产定单 80 台正在生产，2003 年 3 月 10 日完工且未被其他需求占用。这时在对该销售定单进行 MPS 计算时应该将这在制的 80 台按可用量考虑，计算结果为需要生产 20 台计算机。而不是在这 80 台计算机生产完毕入库以后，MPS 计算时，才将其考虑在内。这是 ERP 软件在业务逻辑上容易出错的地方，在实际测试中可以根据上面的描述设计具体的案例。

至此我们的测试用例已基本介绍完毕，在用例的设计过程中我们使用了流程图驱动、等价类划分、边界值分析、因果图等方法。鉴于 ERP 系统的特点，为了方便描述本实例，该用例设计侧重的是其业务流及数据流的连贯性，而数据的有效性校验不作为重点。

在实际设计用例时想要达到 100%覆盖，是不可能完成的任务，而且从成本的角度讲也是行不通的，尤其是像 ERP 这种大型的应用软件，我们应该根据实际的测试目的、测试条件有重点地设计出“适当”的测试用例。



第18章 白盒测试

18.1 综述

本节将提供两个实例，详细地说明白盒测试中如何使用静态测试和动态测试方法，以及使用这两种方法进行测试时相应的测试内容和测试用例的设计过程。

根据白盒测试的特点，一般我们可以在单元测试以及集成测试阶段，使用白盒测试的各种方法，到系统测试阶段，软件的系统结构已经确定，功能接口的黑盒测试将居于主要地位，代码级的白盒测试退居次位：当使用黑盒测试的方法检查错误时，可以通过代码检查定位错误的具体位置，或通过使用性能测试分析性能的瓶颈所在，或通过内存测试，分析出错的原因是否是由于内存使用不当所致。

与其他测试相类似，白盒测试的流程也由测试计划、测试设计、测试实施、测试结果分析等阶段组成。

前面章节中已经介绍了白盒测试中主要使用的静态测试和动态测试方法，下面将分别使用两个例子进行具体说明。

- 在静态测试中将使用一个汽车车身控制的软件示例，测试内容包括静态结构分析和代码质量度量。
- 在动态测试中将使用一个计算器的程序示例，测试内容为覆盖率测试。

18.2 静态测试

汽车车身控制软件用C语言编写，是用于汽车控制的软件，主要包括电动车窗、遥控中央门锁与防盗、外车灯、车内灯、后窗及后视镜加热延时和手动雨刮控制6个模块。

- 测试环境。

① 硬件。普通PC机：

CPU：PIII 600Hz；

内存：128MB；

硬盘：10GB。

② 软件。

操作系统：Windows 2000 Professional 中文版；

编译系统：Visual Studio 6.0。

- 测试工具。

使用 Telelogic 公司的 Logiscope 5.1 作为测试工具。

18.2.1　静态测试结果结构分析

使用测试工具 Logiscope 对上述汽车车身控制软件进行静态结构分析时，可以得到以下几种主要图形。

- 函数调用关系图：以直观的图形方式描述了一个应用程序中各个函数的调用和被调用关系；
- 模块控制流图：显示一个函数的内部逻辑结构；
- 内部文件调用关系图：以图形方式描述了应用程序中各个文件间的调用和被调用关系。

1．函数调用关系图

如图 18-1 所示是汽车车身控制软件的函数调用关系图，图中的每一个方块代表一个函数，方块内的数字代表函数的编号。从图可知，该应用程序从上到下分为 7 层，包括 44 个函数，函数之间没有存在递归调用的情况，结构比较简单。

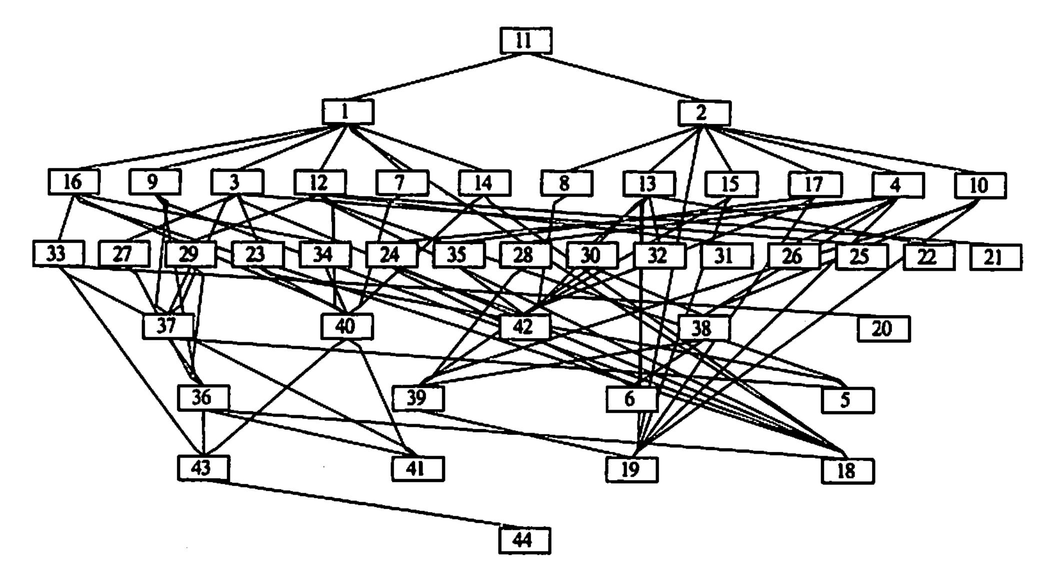


图 18-1　函数调用关系图

44 个函数中编号为 18、19、37、40 和 42 的函数被调用次数较高，其他函数功能的

正确实现严重依赖于这些函数，它们的质量好坏对系统的影响明显要高于其他函数，因此可认为属于重要函数，需要加强对它们的测试，例如使用较高的代码覆盖率要求等措施；其他函数属于次要函数，可以采用较低的代码覆盖率要求。

2．文件调用关系图

文件调用关系图与函数调用关系图相类似，它体现了文件之间的依赖关系。

如图 18-2 所示即为上述汽车车身控制软件的文件调用关系图，其中的方块代表各源文件及其编号，由图可知，该软件包括 6 个源文件，按结构组成可分为 5 层，结构简单。

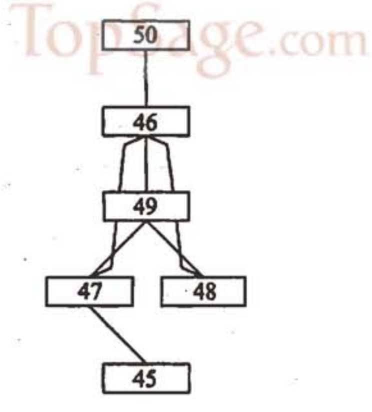


图 18-2 文件调用关系图

3．模块控制流图

如图 18-3 所示是被测汽车车身控制软件中函数 n_Pulse_Counter 的模块控制流图。通过该图可以看出，该函数有一个入口和一个出口，3 个不嵌套的判断结点，结构比较简单，易于实现和理解，属于非常好的结构。

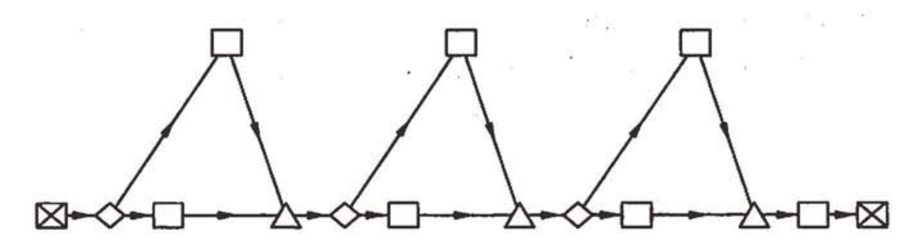


图 18-3 模块控制流图

该汽车车身控制软件的模块控制流图绝大部分与如图 18-3 所示相似，均属于比较好的结构，没有存在重大缺陷。

18.2.2 静态质量度量分析

1．质量度量模型

静态质量度量法通过使用质量度量模型分析程序的复杂性，引用复杂度参数来度量软件是否易于理解、是否可读、是否方便维护、是否可靠。建立的质量度量模型遵循 ISO 9126、DO-178B、SEI/CMM 标准，描述了 Halstead、McCabe 的度量方法学和质量因素（可维护性、可重用性等）和质量准则（可测试性、可读性等），使用多种参数，包括圈复杂度，强化复杂度，设计复杂度，集成复杂度，代码行数，Halstead 复杂度等，例如：可执行语句数、运算符总数、运算符频度、操作数总数、操作数频度、函数中算

子和算法出现频度、函数内调用函数的总数、直接调用函数的数量、调用本函数的上级函数数量、goto 语句数量、类和函数的扇入扇出、注释行的比例、语句函数、平均语句长度、函数参数数量、基本圈复杂度，可以用于全面而系统地评价整个软件系统的质量。

在静态质量度量中，质量度量模型是我们的测试用例。

2．测试结果

上述汽车车身控制软件的静态质量度量结果包括如下内容。

- Metrics Kiviat 图；
- Criteria Kiviat 图；
- Criteria 分布图；
- Factors 分布图。

其中对于程序中的每个函数或方法，我们测量出其所有 Metrics 参数，并使用一个多轴极坐标系来表示，所得到的图称为 Metrics Kiviat 图；按照各 Metrics 参数所属的 Criteria 划分后所得到的 Kiviat 图称为 Criteria Kiviat 图；按照质量模型将软件所有模块的 Criteria 取值分级后，统计各级别模块的比例，用直方图表示，即可得到 Criteria 分布图；同样地按照质量模型将软件所有模块的 Factors 取值分级后，用饼图表示各级别模块的比例，即可得到 Factors 分布图。

（1）Metrics Kiviat 图

如图 18-4 所示是函数 EXT_LMP 的 Metrics Kiviat 图。图中的每一根极轴代表一个 Metrics 参数，所有参数的归一化最小值和最大值组成两个同心圆，当参数的取值位于这两个同心圆之间时，表示满足质量模型要求，否则该参数不符合模型质量要求。

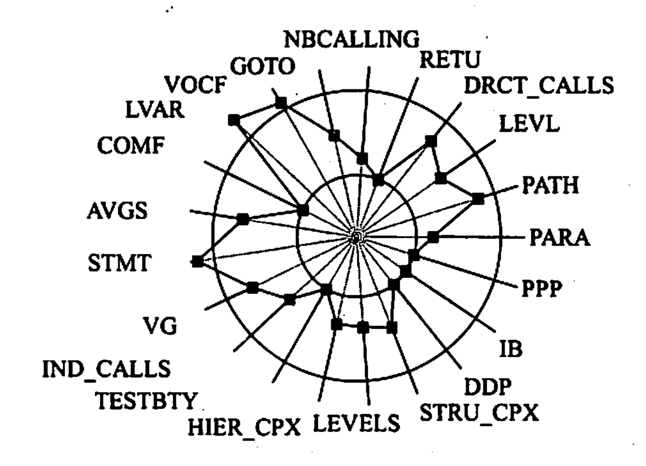


图 18-4　函数 EXT_LMP 的 Metrics Kiviat 图

函数 EXT_LMP 的 VOCF、LVAR 和 STMT 参数值超过标准要求。

其中，VOCF 值过高，意味着该函数包含太近似的，甚至是重复的语句，这样在维护时，由于许多改动可在每处重复的代码处进行，从而引入错误的风险增大，这样的缺陷，可以通过将原代码重复语句分解为可多处调用的子函数来改进。

LVAR 值过高也会对维护带来不利，因为不得不花更多的精力理解这些变量，当对变量理解有误时会带来风险，存在这样的缺陷时，首先要确认局部变量已全部被使用，如果变量数仍过多，应该将函数分解成子函数来处理。

STMT 值过高表示函数内可执行语句较多，经验表明，越多的可执行语句数代表着可执行的操作越多，也意味要花更多的精力去理解该函数，因此对一个函数内的可执行语句数应有限制。这样的缺陷可以采取以下方法解决：将函数分解，提供更好的层次性，提高可维护性。

（2）Criteria Kiviat 图

如图 18-5 所示为函数 EXT_LMP 的 Criteria Kiviat 图，对于每一个函数均可得到一个 Criteria Kiviat 图。

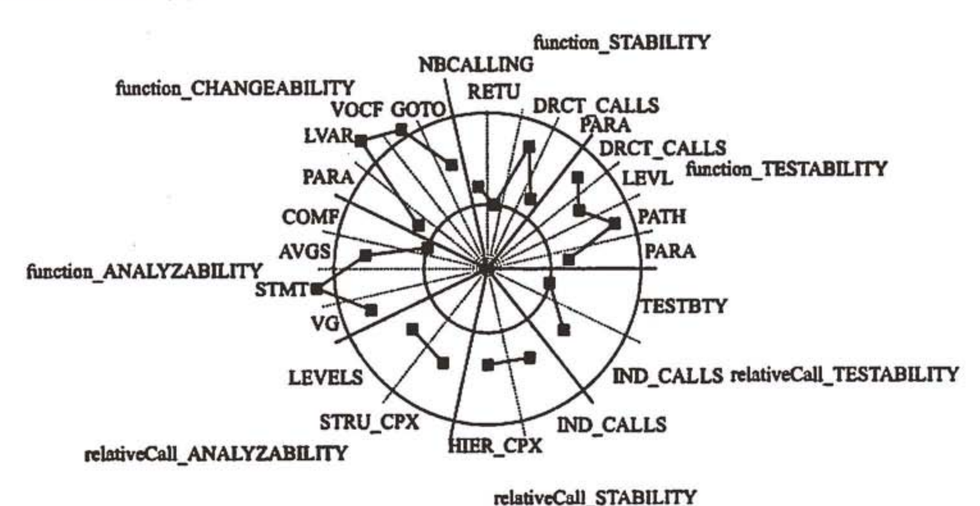


图 18-5 函数 EXT_LMP 的 Criteria Kiviat 图

（3）Criteria 分布图

如图 18-6 (a)、(b)、(c)、(d)、(e)、(f)、(g)所示是 Criteria 层 7 个分类标准：function_TESTABILITY、function_STABILITY、function_CHANGEABILITY、function_ANALYZABILITY、relativeCall_ANALYZABILITY、relativeCall_STABILITY 和 relativeCall_TESTABILITY 的各等级模块分布比例图。

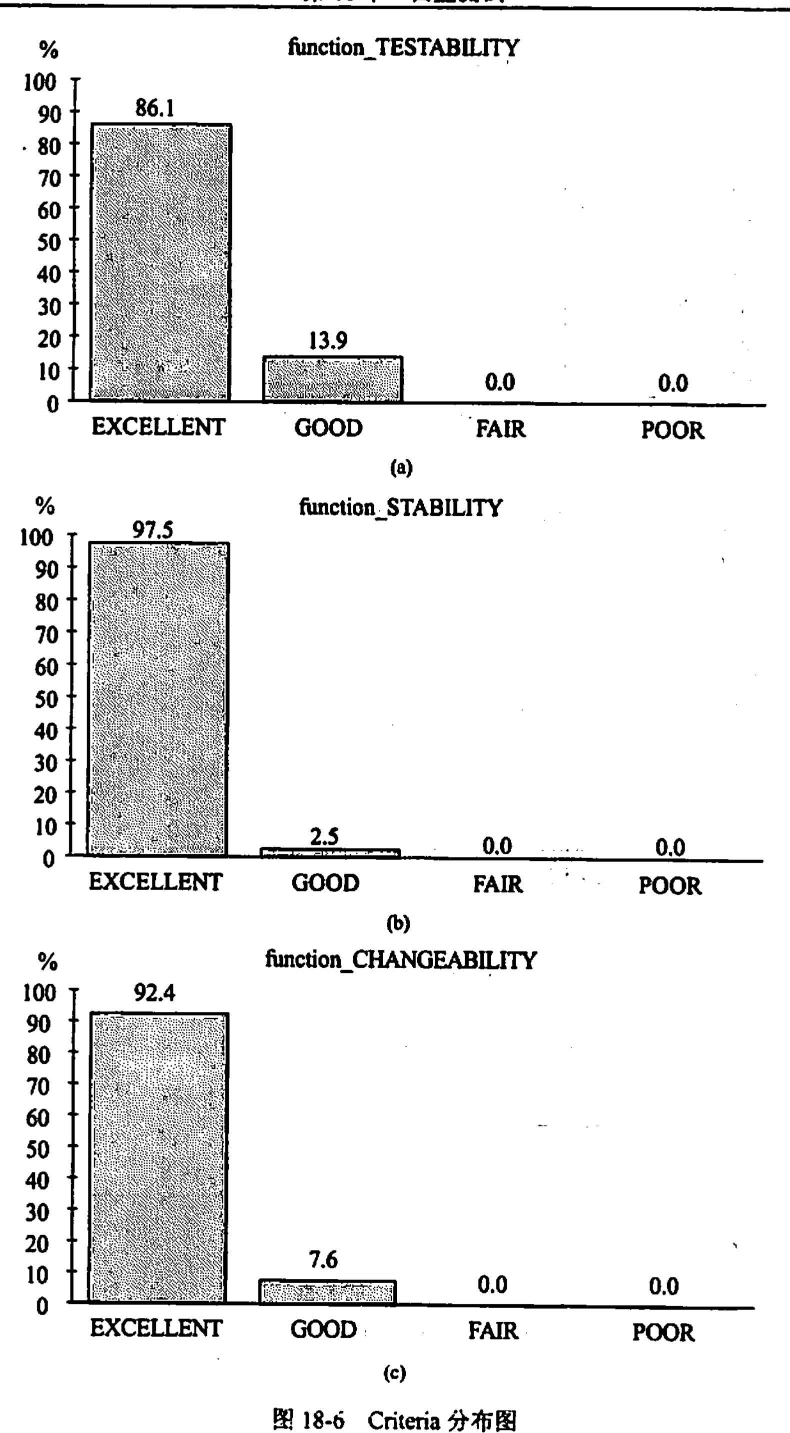
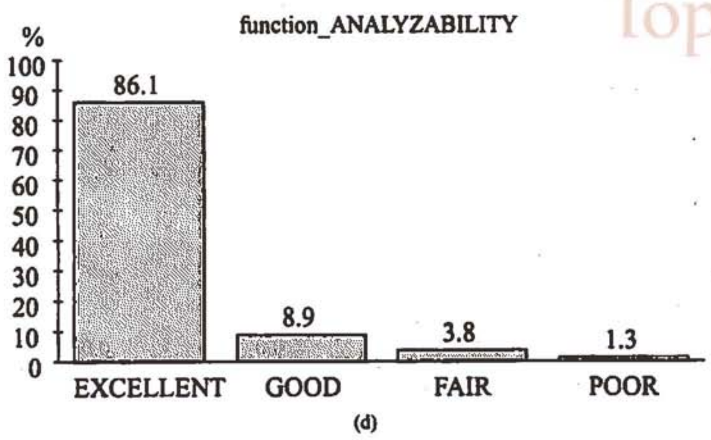
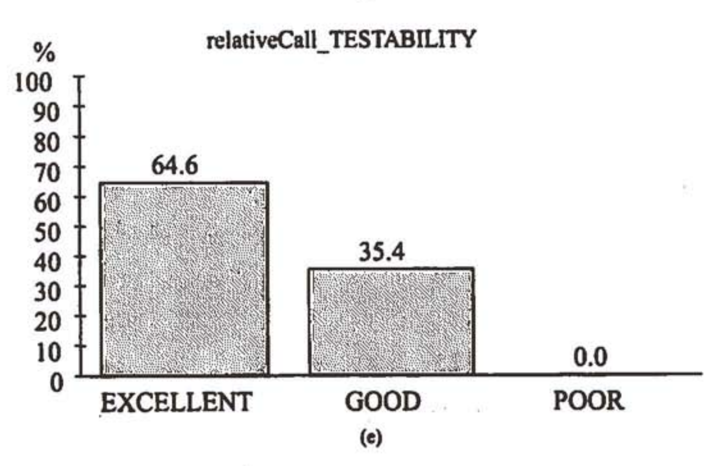


图 18-6　Criteria 分布图

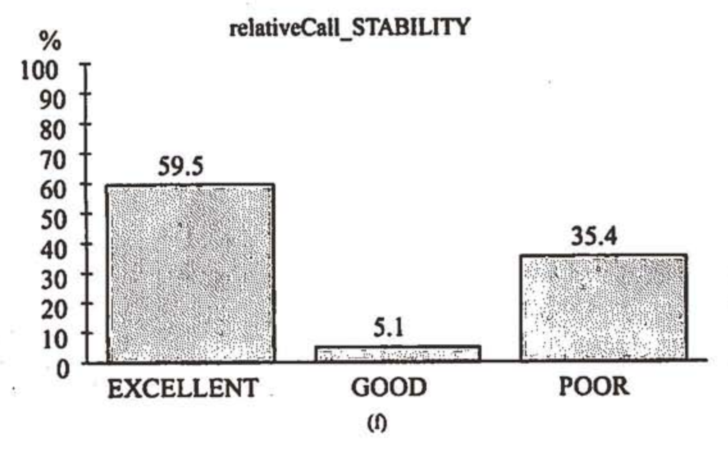

大家网
TopSage.com


function_ANALYZABILITY
%
100
90
80
70
60
50
40
30
20
10
0
86.1
8.9
3.8
1.3
EXCELLENT
GOOD
FAIR
POOR

(d)

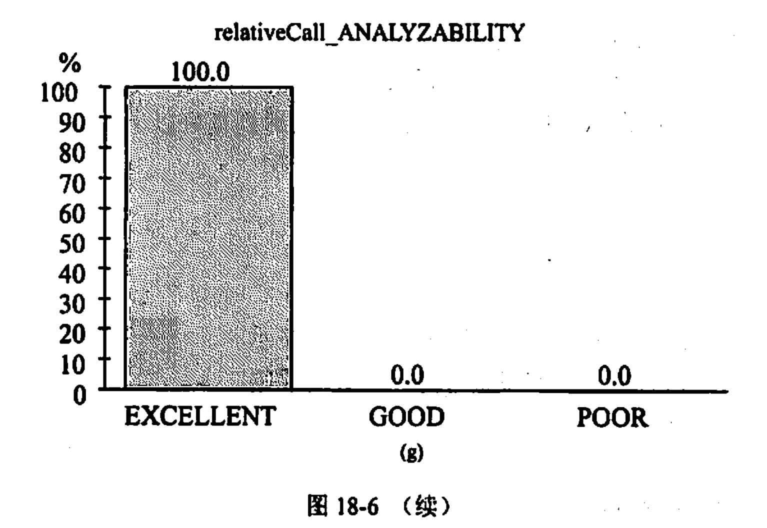

relativeCall_TESTABILITY
%
100
90
80
70
60
50
40
30
20
10
0
64.6
35.4
0.0
EXCELLENT
GOOD
POOR

(e)


relativeCall_STABILITY
%
100
90
80
70
60
50
40
30
20
10
0
59.5
5.1
35.4
EXCELLENT
GOOD
POOR

(f)

图 18-6 （续）



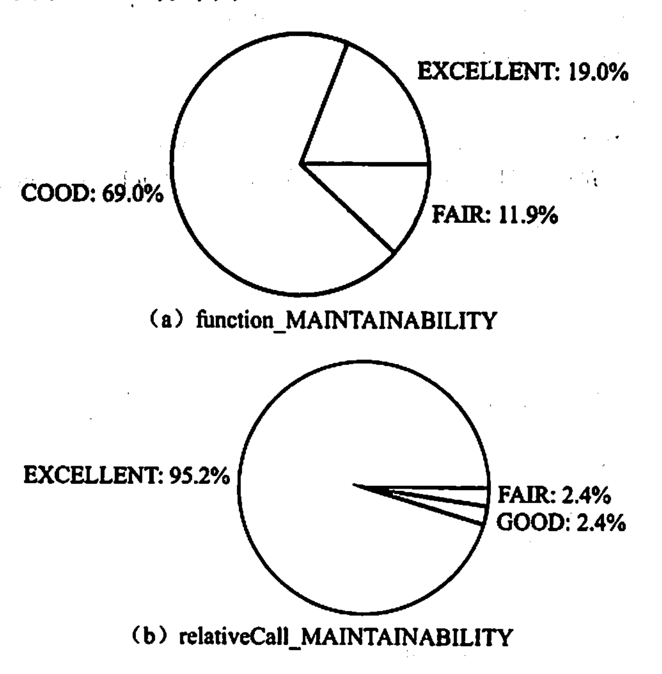
(g)

图 18-6 （续）

从图 18-6 可以看出，该软件 Criteria 层除 relativeCall_Stability 参数外，其他参数属于优良等级的模块均超过了 80%，符合要求。

（4）Factors 分布图

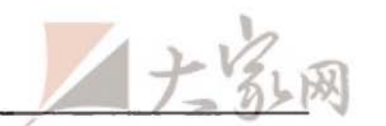
如图 18-7 所示为 Factors 分布图。



（a）function_MAINTAINABILITY

（b）relativeCall_MAINTAINABILITY

图 18-7　Factors 分布图



对于一般的应用软件来说，其 Factors 层属于优良等级的模块应超过 80%，否则，需要根据度量结果进行修改，以满足该要求。该应用软件 Factors 层 function_MAINTAINABILITY 和 relativeCall_MAINTAINABILITY 参数达到良好以上等级的模块超过 80%，满足要求。

18.3 覆盖率测试

在覆盖率测试中我们将使用一个计算器的程序示例，该程序用 C++语言编写，实现整数与浮点数的四则运算功能，其界面如图 18-8 所示。

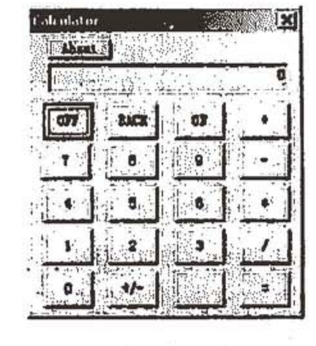


图 18-8 计算器的程序示例界面

- 测试环境。

① 硬件。普通 PC 机；

CPU：PIII 600Hz；

内存：128MB；

硬盘：10GB。

② 软件。

操作系统：Windows 2000 Professional 中文版；

编译系统：Visual Studio 6.0。

- 测试工具。

使用 Applied Microsystems Corporation 公司的 CodeTest 3.5 作为测试工具。

由于篇幅所限，我们将针对其中一个主要的函数“CCacl2Dlg::OnGo()”设计测试用例，该函数响应用户点击按钮“=”的操作，完成计算功能。

该函数代码如下所示。

```
void CCacl2Dlg::OnGo()
{
    if(m_sfmf == TRUE && m_sfms == TRUE && m_sfmfun == TRUE)        //1
    {
        if(m_mfun == 1)                                              //2
            m_result = m_mfir + m_msec;                              //3
        else if(m_mfun == 2)                                         //4
            m_result = m_mfir - m_msec;                              //5
        else if(m_mfun == 3)                                         //6
            m_result = m_mfir * m_msec;                              //7
        else if(m_mfun == 4)                                         //8
```

```
            m_result = m_mfir / m_msec;                                          //9
        m_sfmf = TRUE;                                                           //10
        m_sfms = TRUE;
        m_sfmfun = TRUE;
        m_mfir = m_result;
        m_msec = m_msec;
        m_mfun = m_mfun;
}

else if(m_sfmf == TRUE && m_sfms == FALSE && m_sfmfun == FALSE)                  //11
{
        if(m_fun == 1)                                                           //12
            m_result = m_mfir + m_second;                                        //13
        else if(m_fun == 2)                                                      //14
            m_result = m_mfir - m_second;                                        //15
        else if(m_fun == 3)                                                      //16
            m_result = m_mfir * m_second;                                        //17
        else if(m_fun == 4)                                                      //18
            m_result = m_mfir / m_second;                                        //19
        m_sfmf = TRUE;                                                           //20
        m_sfms = TRUE;
        m_sfmfun = TRUE;
        m_mfir = m_result;
        m_msec = m_second;
        m_mfun = m_fun;
}
else if(m_sfmf = TRUE && m_sfms == TRUE && m_sfmfun == FALSE)                    //21
{
        m_second = m_mfir;                                                       //22
        if(m_fun == 1)                                                           //23
            m_result = m_mfir + m_second;                                        //24
        else if(m_fun == 2)                                                      //25
            m_result = m_mfir - m_second;                                        //26
        else if(m_fun == 3)                                                      //27
            m_result = m_mfir * m_second;                                        //28
        else if(m_fun == 4)                                                      //29
            m_result = m_mfir / m_second;                                        //30
        m_sfmf = TRUE;                                                           //31
```



```
        m_sfms = TRUE;
        m_sfmfun = TRUE;
        m_mfir = m_result;
        m_msec = m_second;
        m_mfun = m_fun;
    }

    else if(m_sfmf == FALSE && m_sfms == FALSE && m_sfmfun == FALSE)          //32
    {
        if(m_EnterSec == FALSE)                                               //33
            m_second = m_first;                                               //34
        if(m_fun == 1)                                                        //35
            m_result = m_first + m_second;                                    //36
        else if(m_fun == 2)                                                   //37
            m_result = m_first - m_second;                                    //38
        else if(m_fun == 3)                                                   //39
            m_result = m_first * m_second;                                    //40
        else if(m_fun == 4)                                                   //41
            m_result = m_first / m_second;                                    //42
        m_sfmf = TRUE;                                                        //43
        m_sfms = TRUE;
        m_sfmfun = TRUE;
        m_mfir = m_result;
        m_msec = m_second;
        m_mfun = m_fun;
    }
    m_x = m_result;                                                           //44
    UpdateData(FALSE);
    m_first = 0;
    m_second = 0.0;
    m_firstz = 0;
    m_firstx = 0.0;
    m_secondz = 0;
    m_secondx = 0.0;
    m_firx = FALSE;
    m_secx = FALSE;
    m_firxw = 0;
    m_secxw = 0;
```

```
        m_firzorf = 1;
        m_seczorf = 1;
        m_ForS = 1;
        m_EnterSec = FALSE;
}
```

在覆盖率测试中，我们按照 100% 路径覆盖的要求设计测试用例，这样可以保证代码中所有的语句得到执行。

18.3.1　测试用例设计

1．以源代码为基础，导出程序的控制流图

根据源代码，我们导出如图 18-9 所示的控制流图。

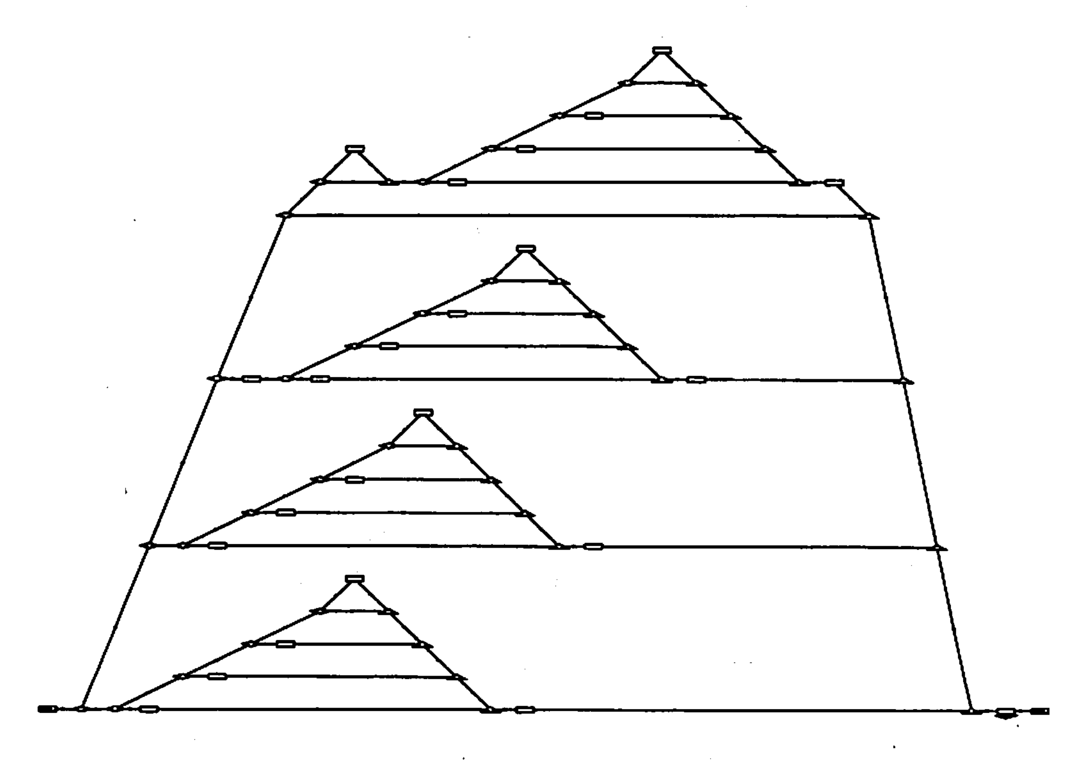


图 18-9　控制流图 1

2．计算得到的控制流图 G 的环路复杂性 V（G）

利用在前面给出的计算控制流图环路复杂性的方法，可以算出：

V(G)=22（区域数）=21（判断结点数）+1=22。

3．确定线性无关的路径的基本集

将如图 18-9 中所示的各结点加入对应编号，得到如图 18-10 所示的控制流图。

图 18-10 控制流图 2

其中结点 1、2、4、6、8、11、12、14、16、18、21、23、25、27、29、32、33、35、37、39 和 41 为判断结点，结点编号与判断表达式对应关系如表 18-1 所示。

表 18-1 结点编号与判断表达式对应关系

结点编号	判断表达式
1	m_sfmf == TRUE && m_sfms == TRUE && m_sfmfun == TRUE
11	m_sfmf == TRUE && m_sfms == FALSE && m_sfmfun == FALSE
21	m_sfmf = TRUE && m_sfms == TRUE && m_sfmfun == FALSE
32	m_sfmf == FALSE && m_sfms == FALSE && m_sfmfun == FALSE
33	m_EnterSec == FALSE

续表

结点编号	判断表达式
2、12、23、35	m_mfun == 1
4、14、25、37	m_mfun == 2
6、16、27、39	m_mfun == 3
8、18、29、41	m_mfun == 4

由如图 18-10 所示可知，22 条线性无关的基本路径为表 18-2 所示的路径。

表 18-2　基本路径

序　号	执行路径
1	1-2-3-10-44
2	1-2-4-5-10-44
3	1-2-4-6-7-10-44
4	1-2-4-6-8-9-10-44
5	1-2-4-6-8-10-44
6	1-11-12-13-20-44
7	1-11-12-14-15-20-44
8	1-11-12-14-16-17-20-44
9	1-11-12-14-16-18-19-20-44
10	1-11-12-14-16-18-20-44
11	1-11-21-22-23-24-31-44
12	1-11-21-22-23-25-26-31-44
13	1-11-21-22-23-25-27-28-31-44
14	1-11-21-22-23-25-27-29-30-31-44
15	1-11-21-22-23-25-27-29-31-44
16	1-11-21-32-44
17	1-11-21-32-33-34-35-36-43-44
18	1-11-21-32-33-35-36-43-44
19	1-11-21-32-33-34-35-37-38-43-44
20	1-11-21-32-33-34-35-37-39-40-43-44
21	1-11-21-32-33-34-35-37-39-41-42-43-44
22	1-11-21-32-33-34-35-37-39-41-43-44

4．生成测试用例，确保基本路径集中每条路径的执行

根据各路径的执行过程，各判断结点取值如表 18-3 所示。

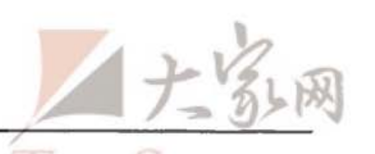


表 18-3 判断结点取值

序号	执行路径	判断结点取值
1	1-2-3-10-44	1：TRUE 2：TRUE
2	1-2-4-5-10-44	1：TRUE 2：FALSE 4：TRUE
3	1-2-4-6-7-10-44	1：TRUE 2：FALSE 4：FALSE 6：TRUE
4	1-2-4-6-8-9-10-44	1：TRUE 2：FALSE 4：FALSE 6：FALSE 8：TRUE
5	1-2-4-6-8-10-44	1：TRUE 2：FALSE 4：FALSE 6：FALSE 8：FALSE
6	1-11-12-13-20-44	1：FALSE 11：TRUE 12：TRUE
7	1-11-12-14-15-20-44	1：FALSE 11：TRUE 12：FALSE 14：TRUE
8	1-11-12-14-16-17-20-44	1：FALSE 11：TRUE 12：FALSE 14：FALSE 16：TRUE
9	1-11-12-14-16-18-19-20-44	1：FALSE 11：TRUE 12：FALSE 14：FALSE 16：FALSE 18：TRUE

续表

序　号	执 行 路 径	判断结点取值
10	1-11-12-14-16-18-20-44	1：FALSE 11：TRUE 12：FALSE 14：FALSE 16：FALSE 18：FALSE
11	1-11-21-22-23-24-31-44	1：FALSE 11：FALSE 21：TRUE 23：TRUE
12	1-11-21-22-23-25-26-31-44	1：FALSE 11：FALSE 21：TRUE 23：FALSE 25：TRUE
13	1-11-21-22-23-25-27-28-31-44	1：FALSE 11：FALSE 21：TRUE 23：FALSE 25：FALSE 27：TRUE
14	1-11-21-22-23-25-27-29-30-31-44	1：FALSE 11：FALSE 21：TRUE 23：FALSE 25：FALSE 27：FALSE 29：TRUE
15	1-11-21-22-23-25-27-29-31-44	1：FALSE 11：FALSE 21：TRUE 23：FALSE 25：FALSE 27：FALSE 29：FALSE



续表

序　　号	执 行 路 径	判断结点取值
16	1-11-21-32-44	1：FALSE 11：FALSE 21：FALSE 32：FALSE
17	1-11-21-32-33-34-35-36-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：TRUE 35：TRUE
18	1-11-21-32-33-35-36-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：FALSE 35：TRUE
19	1-11-21-32-33-34-35-37-38-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：TRUE 35：FALSE 37：TRUE
20	1-11-21-32-33-34-35-37-39-40-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：TRUE 35：FALSE 37：FALSE 39：TRUE
21	1-11-21-32-33-34-35-37-39-41-42-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：TRUE 35：FALSE

续表

序号	执行路径	判断结点取值
22	1-11-21-32-33-34-35-37-39-41-42-43-44	37：FALSE 39：FALSE 41：TRUE
23	1-11-21-32-33-34-35-37-39-41-43-44	1：FALSE 11：FALSE 21：FALSE 32：TRUE 33：TRUE 35：FALSE 37：FALSE 39：FALSE 41：FALSE

根据表 18-3 和如图 18-4 所示，我们可以得到各判断结点中各子条件的取值，如表 18-4 所示（T 代表 TRUE，F 代表 FALSE）。

表 18-4　判断结点与输入条件

序号	判断结点	输入条件
1	1：T	m_sfmf = T && m_sfms = T && m_sfmfun = T
	2：T	m_mfun = 1
2	1：T	m_sfmf = T && m_sfms = T && m_sfmfun = T
	2：F 4：T	m_mfun = 2
3	1：T	m_sfmf = T && m_sfms = T && m_sfmfun = T
	2：F 4：F 6：T	m_mfun = 3
4	1：T	m_sfmf = T && m_sfms = T && m_sfmfun = T
	2：F 4：F 6：F 8：T	m_mfun = 4
5	1：T	m_sfmf = T && m_sfms = T && m_sfmfun = T
	2：F 4：F 6：F 8：F	m_mfun 不等于 1、2、3、4



续表

序　号	判断结点	输入条件
6	1：F 11：T	m_sfmf = F && m_sfms = T && m_sfmfun = T
	12：T	m_mfun = 1
7	1：F 11：T	m_sfmf = F && m_sfms = T && m_sfmfun = T
	12：F 14：T	m_mfun = 2
8	1：F 11：T	m_sfmf = F && m_sfms = T && m_sfmfun = T
	12：F 14：F 16：T	m_mfun = 3
9	1：F 11：T	m_sfmf = F && m_sfms = T && m_sfmfun = T
	12：F 14：F 16：F 18：T	m_mfun = 4
10	1：F 11：T	m_sfmf = F && m_sfms = T && m_sfmfun = T
	12：F 14：F 16：F 18：F	m_mfun 不等于 1、2、3、4
11	1：F 11：F 21：T	m_sfmf = T && m_sfms = T && m_sfmfun = F
	23：T	m_mfun = 1
12	1：F 11：F 21：T	m_sfmf = T && m_sfms = T && m_sfmfun = F
	23：F 25：T	m_mfun = 2

续表

<table>
<tr><th>序　号</th><th>判断结点</th><th colspan="4">输入条件</th></tr>
<tr><td rowspan="2">13</td><td>1：F
11：F
21：T</td><td colspan="4">m_sfmf = T && m_sfms = T && m_sfmfun = F</td></tr>
<tr><td>23：F
25：F
27：T</td><td colspan="4">m_mfun = 3</td></tr>
<tr><td rowspan="2">14</td><td>1：F
11：F
21：T</td><td colspan="4">m_sfmf = T && m_sfms = T && m_sfmfun = F</td></tr>
<tr><td>23：F
25：F
27：F
29：T</td><td colspan="4">m_mfun = 4</td></tr>
<tr><td rowspan="2">15</td><td>1：F
11：F
21：T</td><td colspan="4">m_sfmf = T && m_sfms = T && m_sfmfun = F</td></tr>
<tr><td>23：F
25：F
27：F
29：F</td><td colspan="4">m_mfun 不等于 1、2、3、4</td></tr>
<tr><td rowspan="2">16</td><td rowspan="2">1：F
11：F
21：F
32：F</td><td colspan="4">可使用以下任意一组条件</td></tr>
<tr><td>m_sfmf = F
m_sfms = T
m_sfmfun = F</td><td>m_sfmf = F
m_sfms = F
m_sfmfun = T</td><td>m_sfmf = F
m_sfms = T
m_sfmfun = T</td><td>m_sfmf = T
m_sfms = F
m_sfmfun = T</td></tr>
<tr><td rowspan="3">17</td><td>1：F
11：F
21：F
32：T</td><td colspan="4">m_sfmf = F && m_sfms = F && m_sfmfun = F</td></tr>
<tr><td>33：T</td><td colspan="4">m_EnterSec = F</td></tr>
<tr><td>35：T</td><td colspan="4">m_mfun = 1</td></tr>
<tr><td rowspan="3">18</td><td>1：F
11：F
21：F
32：T</td><td colspan="4">m_sfmf = F && m_sfms = F && m_sfmfun = F</td></tr>
<tr><td>33：F</td><td colspan="4">m_EnterSec = T</td></tr>
<tr><td>35：T</td><td colspan="4">m_mfun = 1</td></tr>
</table>



续表

序　号	判断结点	输入条件
19	1：F 11：F 21：F 32：T	m_sfmf = F && m_sfms = F && m_sfmfun = F
	33：T	m_EnterSec = T
	35：F 37：T	m_mfun = 2
20	1：F 11：F 21：F 32：T	m_sfmf = F && m_sfms = F && m_sfmfun = F
	33：T	m_EnterSec = T
	35：F 37：F 39：T	m_mfun = 3
21	1：F 11：F 21：F 32：T	m_sfmf = F && m_sfms = F && m_sfmfun = F
	33：T	m_EnterSec = T
	35：F 37：F 39：F 41：T	m_mfun = 4
22	1：F 11：F 21：F 32：T	m_sfmf = F && m_sfms = F && m_sfmfun = F
	33：T	m_EnterSec = T
	35：F 37：F 39：F 41：F	m_mfun 不等于 1、2、3、4

18.3.2 测试结果分析

按照如表 18-4 所示设计测试用例并进行测试后，对其代码覆盖结果进行分析，得出如下结果。

- 当完成 100% 路径覆盖时，将同时达到 100% 的 SC、DC、CC 和 CDC；
- 在测试路径 16 时，如果将表中所列的 4 种输入条件都设计出测试用例，还将同时达到 100% 的 MCC 和 MCDC。

第 19 章　数据库测试

19.1　数据库测试概述

19.1.1　数据库系统现状

1．主流数据库系统现状

数据库管理系统经历了 30 多年的发展演变，已经取得了辉煌的成就，发展成了一门内容丰富的学科。目前，市场上具有代表性的数据库产品包括 Oracle、DB2、Sybase 以及 SQL Server 等。在一定意义上，这些产品的特征反映了当前数据库产业界的最高水平和发展趋势。因此，分析这些主流产品的发展现状，是我们了解数据库技术发展的一个重要方面。

数据库管理系统是网络经济的重要基础设施之一。支持 Internet 数据库应用已经成为数据库系统的重要方面。例如，Oracle 公司从 8i 起全面支持互联网应用，是互联网数据库的代表。微软公司更是将 SQL Server 作为其整个.NET 计划中的一个重要的成分。对于互联网应用，由于用户数量是无法事先预测的，这就要求数据库相比以前拥有能处理更大量的数据，以及为更多的用户提供服务的能力，也就是要拥有良好的可伸缩性及高可用性。

数据库技术的广泛使用为企业和组织收集并积累了大量的数据。数据丰富知识贫乏的现实直接导致了联机分析处理、数据仓库和数据挖掘等技术的出现，促使数据库向智能化方向发展。同时企业应用越来越复杂，会涉及到应用服务器、Web 服务器、其他数据库、旧系统中的应用以及第三方软件等，数据库产品与这些软件是否具有良好的集成性，往往关系到整个系统的性能。

基于目前数据库管理系统发展的现状，我国部分数据库专家将数据库技术发展的特点概括为“四高”，即数据库管理系统具有高可靠性、高性能、高可伸缩性和高安全性。数据库是企业信息系统的核心和基础，其可靠性和性能是企业领导人非常关心的问题。随着信息化进程的深化，计算机系统越来越成为企业运营不可缺少的部分，这时，数据库系统的稳定和高效是必要的条件。在互联网环境下还要考虑支持几千或上万个用户同时存取和 7×24 小时不间断运行的要求，提供联机数据备份、容错、容灾以及信息安全措施等。

2. 我国数据库系统的发展现状

数据库管理系统是国家信息基础设施的重要组成部分，是我国民族 IT 产业及软件产业发展的支撑技术。多年来，我国一直非常注重开发自主产权的数据库产品，科技部对国产数据库软件产品的开发给予了特别的支持。

目前，我国数据库系统已经具备了普通应用所需的基本数据库管理功能，也具有一定的扩展性，能够满足一般的应用需求。但国产数据库系统在易用性和可管理性方面，相比于国外主流数据库系统，还存在一定的差距，同时，国产数据库系统运行的稳定性，执行效率，特别是在高并发、多用户环境等复杂环境下的执行性能和稳定性，同国外主流数据库相比还有一些差距。而且，国产数据库系统在实际的信息系统中得到应用的机会并不多，要投入大规模的企业应用，还有待实践的考验。

19.1.2 数据库系统评测体系

就数据库领域测试而言，目前国际上的 TPC 组织虽然提出了性能测试标准，然而由于数据库应用的领域不同、运行的环境不同，加上数据库管理系统功能、结构日趋复杂、度量特性繁多等诸多因素，目前国际上并没有被普遍接受的数据库综合评测体系。

2002 年开始至今，中国软件评测中心与清华大学软件学院共同承担了国家 863 计划“数据库管理系统测试及其工具研发”课题，该课题的主要目的就是围绕国产大型通用数据库管理系统的开发、应用、推广的全过程，研发有效、实用的数据库管理系统测试工具与环境，建立国家数据库管理系统评测体系，为政府部门（科技部）数据库管理系统专项择优提供技术依据，为数据库管理系统开发商、数据库应用系统开发商和数据库系统的最终用户提供质量服务，形成服务社会的公共数据库管理系统专业测试平台，从而促进国产数据库管理系统的产品化、产业化进程。

课题研究、实施期间，在国家科技部任命的数据库专家组的指导下，两家课题承担单位深入地研究了国内外软件测试技术的最新进展，针对我国数据库管理系统的发展现状进行了深入的调查研究，恰当地选择了相关的国际标准。在课题实施的技术路线方面，针对数据库管理系统测试工作中亟待解决的关键问题进行了重点攻关，主要包括相关国际标准的研究、测试平台的设计、构造测试用例和制定测评标准等；在借鉴已有国内外数据库管理系统成功经验的基础上，结合我国数据库管理系统研发的现状进行创新，探索高效的有利于促进我国数据库管理系统良性发展的数据库测试技术、方法与模式；重点考虑了规范化工作和资源合理搭配，卓有成效。形成了一套相对完整的数据库管理系统评测体系。

该体系参照数据库相关国际标准和有关软件系统测试规范，配合国产数据库管理系统的开发进程，将数据库管理系统测试分解为四个方面：产品确认测试、标准符合性测



试、基准性能测试和应用综合测试。通过综合以上四个方面的测试，完成对数据库管理系统的全面评测。具体包括以下几个方面。

- 产品确认测试。

按照 GB/T 16260《软件产品质量评价特性及应用指南》、GB/T 17544《软件包质量要求和测试》的相关标准，参考数据库管理系统开发商提供的文档资料，重点测试数据库管理系统的扩展性、可靠性、安全性、大数据量、管理工具、用户文档六个方面，以度量数据库管理系统的产品化程度，在此过程中形成了一套比较完善的确认测试用例集。

- 标准符合性测试。

① SQL 标准符合性测试：按照 SQL92 标准，全面测试一个数据库产品的 SQL 标准支持特性。

② ODBC 标准符合性测试：采用 SWsoft Inc 开发的 ODBC 2.5 标准符合性测试工具进行测试。在此基础上按照 ODBC 3.0 标准对测试用例进行了相当规模的修改和扩充，并且将微软公司的 QuickTest 测试工具的部分模块集成到该测试工具中。

③ JDBC 标准符合性测试：在 SUN 公司开发的 JDBC 标准符合性测试工具的基础上，按照 JDBC 3.0 标准和国产数据库研发的具体要求，对测试用例进行了修改和扩充。

- 基准性能测试。

① TPC-C 测试：按照 TPC-C 标准自主开发了相应的测试工具。该工具采用 SEDA 型架构，可适用于高并发度的服务器程序，满足 TPC-C 测试要求的服务器负荷要求。

② TPC-W 测试：按照 TPC-W 标准，参照 wisconsin 测试工具作了大量修改后形成了自己的测试工具。

如何根据我国数据库管理系统的发展水平，恰当地选用数据库相关技术与标准是顺利有效地开展测评工作的关键，也是难点。我们努力做到既符合国产数据库管理系统发展水平，又引导国产数据库管理系统与主流数据库技术接轨。在课题实施过程中完善了数据库管理系统综合测评体系；结合我国数据库领域多年积累的技术与实践知识，完善了符合我国实际情况的数据库管理系统测评标准。

19.2 产品确认测试

19.2.1 系统功能测试

1．概述

以数据库系统“四高”技术要求为指导思想，在国家 863 计划数据库专家组的指导下，中国软件评测中心及清华大学软件学院结合自己的实践经验，在参考国外主流数据

库产品功能特性的基础上，制定了国产数据库管理系统产品确认测试的评测要点，其主要从数据库产品的可扩展性、可靠性、安全性、大数据量、系统功能、用户文档六个方面，形成对数据库基本功能的全面评估与测试。其中，以系统功能为主要测试对象，因为能否正确地提供数据存储及管理的功能、能否有效和正确地对存储在数据库中的模式对象和非模式对象进行管理，是其能否正常提供数据管理服务的基础，也是数据库管理系统能够真正投入使用的前提，同时考虑到数据库系统的可管理性，这些系统功能均通过图形化管理工具来测试。

2．测试内容

经过认真的讨论，在多次反复征求相关意见的基础上，测试组最终确定系统功能部分的测试点为安装与配置、数据库存储管理、模式对象管理、非模式对象管理、交互式查询工具、性能监测与调优、数据迁移工具及作业管理几个方面。各个部分又分成若干个具体的测试项目，具体测试点概括如下。

- 安装与配置：主要测试数据库管理系统是否具有完整的图形化安装程序；是否提供集中式多服务器管理及网络配置；是否在安装界面中显示数据文件、日志文件、控制文件等参数文件的默认路径及其命名规则；以及是否提供运行参数查看与设置功能；能够正确地进行数据库的创建和删除等。
- 数据库存储管理：主要测试点为表空间（文件组）管理、数据文件管理、日志文件管理以及归档文件管理等功能。
- 模式对象管理：模式对象管理是数据库管理系统最基本的数据管理服务功能特性，是数据库所有功能的基础。其主要测试功能点包括表管理、索引管理、视图管理、约束管理、存储过程管理和触发器管理等。

① 表管理：主要测试点为图形方式下表的创建，图形方式下修改表、数据类型下拉框选择与修改、重组表数据、图形工具中查看编辑数据，支持图形下拉框条件选择与查询、表属性及相关性图形化显示等。

② 索引管理：主要测试点为创建、修改索引信息，提供索引定义类型选择，索引的存储管理，索引重组与合并等。

③ 视图管理：主要测试点包括图形方式下创建、删除视图，图形工具中查看视图定义，图形工具中查看视图数据、支持条件查询，视图属性及其相关性图形化显示等。

④ 约束管理：主要测试点包括约束定义与修改（主键/外键/（NOT）NULL/CHECK/UNIQUE/DEFAULT 设置）、支持约束状态控制（延时/立即）、约束查看、相关性图形化显示等。

⑤ 存储过程管理：主要测试点包括创建、删除存储过程；图形工具中查看、修改存储过程代码（支持所得即所写）等。



⑥ 触发器管理：主要测试点包括支持图形工具中创建、删除触发器；支持行级触发器；支持语句级触发器；图形工具中查看、修改触发器代码（支持所得即所写）等。

- 非模式对象管理：主要测试点为模式管理，包括模式的创建、删除、查看、用户指派等；用户管理，包括用户的创建、删除、修改、授权、口令策略管理；角色管理，包括角色的创建、删除、修改、查看、用户指派；权限管理，包括数据库对象权限的查看与指派、用户对象权限的查看与指派；审计选项设置，包括语句审计、对象审计、权限审计、审计开关等。
- 交互式查询工具：主要测试点包括易用性、稳定性等。
- 性能监测与调优：要求以图形方式提供 SQL 语句执行计划、提供数据库运行图形监控、提供可配置的性能数据跟踪与统计、提供死锁监测与解锁功能等。
- 数据迁移工具：要求支持 TXT 文件的数据迁移；支持 EXCEL 文件的数据迁移；支持 XML 数据导出；支持从 SQL Server 的表、约束及数据迁移；支持从 Oracle 的表、约束及数据迁移；支持从 DB2 的表、约束及数据迁移以及从 Oracle 进行数据迁移的性能等。
- 作业管理：包括作业调度、通知（操作员）管理、维护计划管理等。

3．测试方法

采用黑盒测试方法，主要通过图形化管理工具、交互式 SQL 工具等对数据库管理系统的功能特性进行测试。要求被测数据库提供 Windows 和 Linux 平台上的图形化管理工具，任一平台上的工具都能够管理 Windows 和 Linux 平台上的数据库服务器。由于该部分测试为功能验证性测试，因此以手工测试为主。

4．测试用例设计

针对每一个测试点，我们均设计相应的测试用例对其进行考察，测试用例包括输入、预期的结果以及功能是否正确的判断标志，由于受篇幅的限制，仅以表数据重组为例进行说明。如表 19-1 所示（表数据重组功能点测试用例说明）。

表 19-1 表数据重组测试用例

序号	输入	预期结果	判断标志
1	①设立源数据库，准备原始数据 ②管理系统为Oracle，数据库名称为CCID，表名为：Sys_ProdOrder_MachineCode，共522252条记录，其中，DeletedFlag1='Y'的记录约为20万，DeletedFlag2='Y'的记录约为15万，且这35万记录随机分布，互不重复	在 Oracle 数据库中生成相应的数据表	
2	用以下语句建立目的表（注意按需改动数据类型） CREATE TABLE Sys_ProdOrder_MachineCode (Code char (12) NOT NULL ,	在被测数据库中生成相应的数据表	

续表

序　号	输　入	预期结果	判断标志
2	MachineCode char (32) NOT NULL , NewMachineCode varchar (32), ProdPos varchar (50), ProdLine varchar (36) , Amount int, Price float, InputDate　date, Descript　varchar (254), OrderId int, DeletedFlag1 char(1), DeletedFlag2 char(1), Primary key (MachineCode))	在被测数据库中生成相应的数据表	
3	执行数据导入	将源数据库中表和数据正确导入到被测数据库系统中	
4	执行数据删除 DELETE FROM Sys_ProdOrder_MachineCode WHERE DeletedFlag1='Y';	批量删除数据成功	
5	执行第一次查询，并记录查询时间为 T1 SELECT count(*) FROM "CONTEST"."SYS_PRODORDER_MACHINECODE";	重组前进行数据查询	
6	进行表数据重组	表重组成功	
7	表数据重组后再查询，并记录查询时间为 T2 SELECT count(*) FROM "CONTEST"."SYS_PRODORDER_MACHINECODE";	重组后进行数据查询	如果 T1>T2 则表明数据重组有效，该测试项的结果为通过

19.2.2　可靠性测试

1．测试内容

作为支撑企业应用的后台核心和基础，数据库系统的稳定可靠性是应用企业最关心的问题，它与整个企业的经营活动密切相关，一旦出现宕机或者数据丢失，企业的损失将无法估量，这不仅仅只是企业的经济利益会遭受损失的问题，甚至会引起一些法律纠纷，比如银行系统和证券系统等。另外，在一些意外造成数据库服务停止的情况下，如



何尽快地恢复服务也是必须考虑的问题。因此，决定对数据库系统 7×24 小时不间断运行的能力、数据备份、容错、容灾等能力进行测试，并确定测试点如下。

- 数据库备份：考察数据库系统能否支持多种完全备份方式，包括对指定库、指定某一对象、指定一组对象进行备份；是否支持多种完全还原方式，包括对指定库、指定某一对象、指定一组对象进行还原；是否支持对指定库进行增量备份以及支持联机备份等，同时还有考察系统备份恢复的效率。
- 故障恢复：在系统出现故障或者存储介质出现故障的情况下，数据库系统是否提供相应的数据恢复机制。
- 运行稳定性：数据库系统的长期稳定运行能力是应用系统对后台数据库的最基本的要求，因此有必要对数据库进行不间断运行 7×24 小时的测试。
- 数据库复制：数据库系统是否提供了数据库复制的机制。

2．测试方法

采用黑盒测试方法，对 Windows 和 Linux 平台上的数据库服务器分别进行测试。由于该部分测试为功能验证性测试，因此以人工测试为主，同时，借助 TPCC 测试程序产生各种工作负载并进行结果验证。

3．测试用例设计

针对每一个测试点，我们都设计了完整的测试用例对其进行考察，测试用例包括输入、预期的结果以及功能是否正确的判断标志，在本部分中，我们充分利用了 TPCC 测试程序联机事务处理的特点，在运行 TPCC 测试程序的同时，完成联机备份、故障恢复等测试项的测试工作。由于 TPCC 测试程序可以加载不同的数据量，借助该工具我们也可以完成对数据库完全备份、增量备份的测试工作。TPCC 测试程序的最大特点就是频繁的联机事务处理，因此它对后台数据库的稳定运行也有较高的要求，利用该工具，我们可以完成对数据库系统 7×24 不间断运行能力的测试。

19.2.3 安全性测试

1．测试内容

数据库的安全性主要是指数据库的用户认证方式及其权限管理，当数据库遭受非法用户访问时，系统的跟踪与审计功能等。具体的测试点如下。

- 用户及口令管理。包括用户定义与管理、角色定义与管理、口令管理等。
- 授权和审计管理。主要测试点为数据库审计、授权管理（表权限/列权限）、支持操作系统用户验证方式等。

2．测试方法

采用黑盒测试方法，对 Windows 和 Linux 平台上的数据库服务器分别进行测试。由

于该部分测试为功能验证性测试，因此以人工测试为主。

3．测试用例设计

针对每一个测试点，我们都设计了完整的测试用例，对其进行考察，测试用例包括输入、预期的结果以及功能是否正确的判断标志。

19.2.4 扩展性测试

1．测试内容

企业信息系统投入使用以后，采集的数据越来越多，同时企业的业务量也在不断增长，从而对信息系统的升级变得越来越迫切，因此数据库系统的可扩展能力也显得越来越重要。由于业务的扩大，原来的系统规模和能力已经不再适应新的要求的时候，不是更换更高档次的机器，而是在原有的基础上增加新的设备，如处理器、存储器等，从而达到分散负载的目的。为了对数据库系统的这些特性进行测试，具体测试点为：

- 数据库系统的跨平台支持。
- CPU 加速比。

2．测试方法

对于数据库系统的跨平台支持测试，采用黑盒测试方法，对被测软件在各种平台上的样品进行安装测试，并验证主要功能。

对于 CPU 加速比的测试，采用了 TPCC 测试工具来进行，分别测试不同 CPU 数环境下的 TPMC 值，计算出相应数据库的 CPU 加速比值。在具体的测试过程中其测试方法完全等同于 TPCC 的测试方法。

3．测试用例设计

数据库系统的跨平台支持测试的安装平台主要包括 Windows、Linux 和 Solaris 平台。

对于 CPU 加速比的测试，首先在两个 CPU 的服务器进行了 TPCC 测试，取得该环境下的最大 TPMC 值，然后，在保持其他条件不变的情况下，将服务器的 CPU 个数增加到 4 个，并取得了 4 个 CPU 下的最大 TPMC 值，最后得到相应数据库的 CPU 加速比值。

19.3 标准符合性测试

19.3.1 SQL 符合性测试

1．概述

SQL（Structured Query Language）于 1974 年提出，1986 年 10 月成为数据库语言的美国标准，1987 年成为 ISO 标准，以后多次升版，目前市场普遍接受的是 SQL92 标准。

所谓 SQL 标准符合性测试即测试数据库管理系统与 SQL 标准的符合程度，分为语法级（即所接受的 SQL 语言与标准 BNF 的符合程度）、语义级（即数据库管理系统所实现 SQL 语言执行结果与 SQL 标准规定结果的符合程度）。目前美国标准技术研究所（NIST）的 SQL 测试用例库代表了符合性测试的主流用例库，数据库系统对该测试用例库的支持程度是数据库管理系统标准化的重要指标，是数据库管理系统进入市场的基础，也是用户（包括最终用户和应用开发商）选择数据库管理系统的重要依据之一。

2. 测试内容

本测试的测试用例参照美国 NIST 的 SQL Test Suite Version 6.0 嵌入式 C 和交互式的入门级和过渡级部分，关于过渡级测试用例在本测试中仅采用子集。有关的测试用例集文件可以从网址 http://www.itl.nist.gov/div897/ctg/sql_form.htm 获取，同时，NIST 提供的测试用例补丁下载网址为 http://www.itl.nist.gov/div897/ctg/sql-testing/upd600.ted，在整个测试过程中，只需要执行全部的测试用例文件，最后统计通过的测试用例即可。

3. 测试工具

SQL 标准符合性测试采用黑盒测试方法，在详细研究美国标准技术研究所（NIST）的测试用例库的基础上，测试组自行开发了集测试和结果定量分析于一体的自动化测试工具，利用该测试工具可以直接选择被测文件来运行并统计相应的测试结果。

4. 测试过程

安装测试工具后，在程序安装路径下生成 CaseDescribe、Help 和 TestCase 三个文件夹，分别用于存放测试用例描述、程序帮助文档和测试用例。

测试中，首先在基本配置选项卡进行数据库连接配置，并进行数据库连接测试。然后在嵌入式/交互式测试配置选项卡上对指令格式和出错关键字进行配置。指令格式中采用符号“*”表示文件名，出错关键字可以使用符号“&&”和“||”表示与和或的关系。

数据库连接配置成功后，即可选择测试用例列表文件和测试用例所在文件夹进行测试，测试前要在测试选项卡界面的测试级别复选框中选中要测试的测试级别。

测试过程中，测试选项卡界面文件名列表的左侧以图标方式表示执行情况，右侧的空白区域显示相应操作的输出信息。

测试完成后，对于嵌入式测试在结果统计选项卡上进行结果统计。对于交互式测试，测试结果保存在生成的结果文件中。

5. 测试结果

NIST 提供的测试用例集包含了若干测试用例文件，每个测试用例中包含若干个测试点，任何一个测试点出现错误均认为该测试用例没有通过，其中测试用例的执行结果以 HU 用户的 TestReport 中记录的结果为准。

通过的入门级测试用例数占入门级测试用例总数的比例，即为入门级测试通过率。通过的过渡级测试用例数占过渡级测试用例总数的比例，即为过渡测试通过率。

19.3.2 ODBC符合性测试

1．概述

开放数据库互连（Open DataBase Connectivity，ODBC）是微软公司开发的一套开放数据库系统应用程序接口规范，它是微软公司 Windows 开放系统体系结构（Windows Open System Architecture，WOSA）的主要组成部分。ODBC规范为应用程序提供了一套高层调用接口规范和基于动态链接的运行支持环境。使用ODBC开发数据库应用程序时，应用程序调用的是标准的ODBC函数和SQL语句，数据库的底层操作由各个数据库的驱动程序完成。所以这样的应用程序具有很好的适应性和可移植性，并且具备同时访问多种数据库系统的能力。

2．测试内容

本测试参照Microsoft ODBC 3.0标准进行。ODBC标准符合性测试的具体项目包括：API函数和数据类型支持性测试、数据库连接功能测试、基本功能测试、元数据功能测试、诊断函数功能及错误状态码测试、游标功能测试、事务功能测试、标量函数测试、ODBC SQL语法测试等。

3．测试工具

ODBC标准符合性测试以SWsoft Inc开发的ODBC 2.5标准符合性测试工具为原型，按照ODBC 3.0标准对其进行了较多的修改和扩充，将微软的QuickTest测试工具的部分模块集成到该测试工具中，并且加入了测试结果的定量分析功能。

4．测试过程

安装完测试工具后，首先对ODBC数据源进行配置，设置被测数据库的连接驱动并进行连接测试。

连接测试成功后，即可选择测试结果信息的输出文件，最后选中相应的测试项目或者全部测试项目，单击“开始”按钮进行测试。

在本测试中，主要的测试方法为根据所设的ODBC版本信息，通过SQLGetFunction取出并显示厂商声明支持的API函数（包括版本信息），其后参照微软提供的ODBC测试工具（微软的测试工具仅对每个函数选定了一种最简单的参数组合来测试）验证驱动程序实际的支持性，然后再通过其他更详细、更复杂的测试用例来重点验证应用中比较重要的一部分函数，考察其支持的程度。对于数据类型的测试，首先根据所设的ODBC版本信息，通过 SQLGetTypeInfo 取出并显示厂商声明支持的所有数据类型，然后通过ODBC元数据创建相关的表来测试实际支持的基本数据类型，包括大字段的测试。



5．测试结果

API 函数的测试，参照微软的测试工具（QuickTest）对每个函数选定了一种最简单的参数组合来测试，仅用其作简单的支持性测试。此项测试根据通过测试的函数的百分比来计算。对于其他的更重要的应用功能是通过其他更详细、更复杂的测试用例来验证的，其执行结果的成功与否直接记录为测试结果。

19.3.3 JDBC 符合性测试

1．概述

JDBC 规范为 Java 语言访问关系数据库提供了一个编程接口规范，它由一组用 Java 编程语言编写的类和接口组成。JDBC 规范为数据库开发人员提供了一个标准的 API，使他们能够用纯 Java API 来编写数据库应用程序。目前 JDBC 规范的最新版本为 JDBC3.0 规范。通过对以上标准和规范的符合性测试，体现了数据库管理系统对应用开发的支持水平。

2．测试内容

根据 JDBC 规范 3.0 符合性要求，对其规定的接口和类进行测试，由于 SUN 公司提供的 JDBC API Testsuite 1.3.1 工具中覆盖了大部分的接口和 API 函数的测试用例，但还有部分规范中有要求，但该测试工具没有测试到的项目，这些测试项目由测试组添加到测试用例中。具体包括 JDBC 3.0 规范 DataBaseMetaData 接口中新增的方法、JDBC 3.0 规范新增的接口 ParameterMetaData 和 SavePoint、JDBC 3.0 规范中接口 BLOB 和 CLOB 的 API 函数、JDBC 3.0 规范中其他接口中与新增接口有关的 API、JDBC 3.0 中定义的对 autogenerated keys 特征的支持、结果集的游标保持特性等。

3．测试工具

测试工具采用 SUN 公司提供的 JDBC API Testsuite 1.3.1 为基本测试工具。测试时服务器端为 Windows 平台和 Linux 平台，客户端运行在 Windows 平台，Java 运行环境采用 J2SDK1.4 和 J2SDKEE 1.3.1。该测试工具在 SUN 公司开发的 JDBC 标准符合性测试工具的基础上，按照 JDBC 3.0 标准对测试用例进行了修改和扩充，并且加入了测试结果的定量分析功能。修正之前，原始的测试工具及相关工具可以从以下地址获取：http://java.sun.com/products/jdbc/download.html；http://java.sun.com/products/jdbc/download.html#testsuite131。

4．测试过程

安装完测试工具后，首先对 Java 运行环境进行设置，然后对数据库 URL（含数据库名）、DDL 文件、DML 文件等进行配置。配置结束后即可开始具体的测试工作。

5．测试结果

JDBC 标准符合性测试对数据库系统分别在 Windows 和 Linux 操作系统上的符合性进行测试，测试完成后，统计各个接口或类中 API 函数通过的测试用例点的数量，按用

例通过的比例计算其对标准的符合程度。

19.4 系统性能测试

19.4.1 概述

就数据库测试而言，国际上 TPC 组织提出的性能测试标准和规范，是最为大家所熟悉的数据库测试规范。TPC 是交易处理性能委员会（Transaction Processing Performance Council）的缩写，该组织最早成立于 1988 年，由一些在计算机领域提供软硬件系统或者相关解决方案的厂商组成，会员从成立之初的 8 家公司发展到目前的 50 余家，IBM、NCR、HP、Oracle、Microsoft 等国际著名公司均是其会员。

TPC 是一个非盈利性机构，其目的主要是为了针对特定的领域，如联机交易处理系统（On Line Transaction Processing，OLTP）、数据仓库或决策支持系统（Decision Support System，DSS）、电子商务解决方案等，制定相应的性能测试规范，从而为用户在选择相应解决方案的平台时提供参考标准。

TPC 组织制定的数据库评测规范主要包括 TPC-A、TPC-B、TPC-C、TPC-D、TPC-H/TPC-R、TPC-W 等，而目前正在使用的性能评测规范有以下几种。

- 针对 OLTP 系统的性能评测规范 TPC-C。
- 针对数据仓库或决策支持系统的性能评测规范 TPC-H 和 TPC-R：前者用于评估数据仓库系统中的动态查询负载，后者用于评估数据仓库系统中的预定义报表应用，两者均源自早期的 TPC-D。
- 针对电子商务应用的性能评测规范 TPC-W。

如果使用恰当，这些测试规范对于用户在选择相应的系统平台时，可以起到较好的参考作用。

本次数据库测试，我们对数据库系统进行了 TPC-C 和 TPC-W 测试，由于这两类测试的测试规范是成型的，因此，最重要的是依据相应的测试规范去建立测试模型、配置数据库参数、开发相应的测试程序或者工具，当然，具体的程序开发不在本书所要阐述的内容范围之内，因此，我们所要讲述的重点在于，如何去理解这些标准和规范，以及如何去正确的理解和分析进行 TPC-C 和 TPC-W 测试后产生的结果数据。

19.4.2 TPC-C 测试

1．规范概要

TPC-C 规范是专门针对联机交易处理系统（OLTP 系统）的，一般情况下我们也把



这类系统称为业务处理系统。这类系统具有比较鲜明的特点，这些特点主要表现如下。

- 多种事务处理并发执行，充分体现了事物处理的复杂性；
- 在线与离线的事务执行模式；
- 多个在线会话终端；
- 适中的系统运行时间和应用程序运行时间；
- 大量的磁盘 I/O 数据流；
- 强调事务的完整性要求（ACID）；
- 对于非一致的数据分布，使用主键和从键进行访问；
- 数据库由许多大小不一、属性多样，而又相互关联的数据表组成；
- 存在较多数据访问和更新之间的资源争夺。

为此，TPC-C 测试规范中模拟了一个比较复杂，并具有代表意义的 OLTP 应用环境，来对数据库管理系统的联机事务处理性能进行测试。

2．测试模型

TPC-C 测试用到的模型是一个大型的商品批发销售公司，它拥有若干个分布在不同区域的商品仓库。当业务扩展的时候，公司将添加新的仓库。每个仓库负责为 10 个销售点供货，每个销售点为 3000 个客户提供服务，每个客户提交的订单中，平均每个订单有 10 项产品，所有订单中约 1%的产品在其直接所属的仓库中没有存货，必须由其他区域的仓库来供货。同时，每个仓库都要维护公司销售的 100 000 种商品的库存记录。如图 19-1 所示。

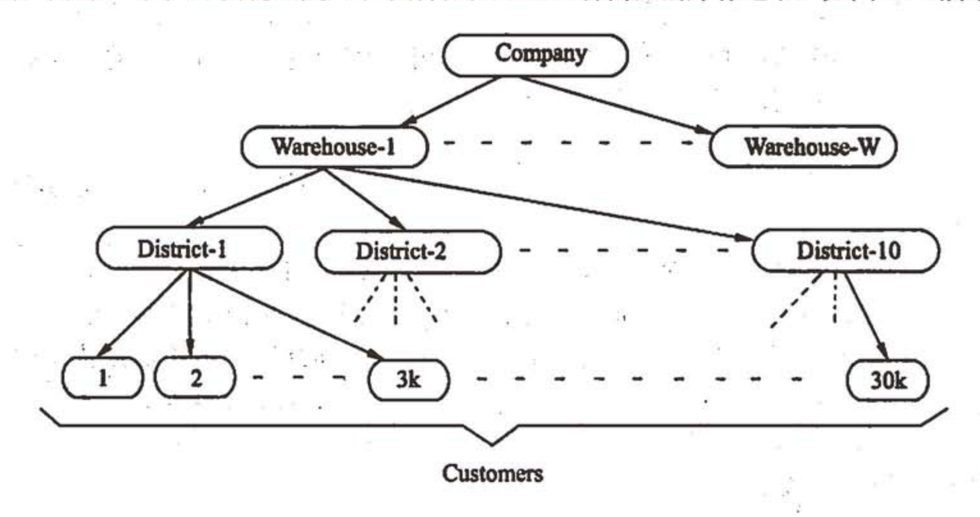


图 19-1 TPC-C 测试用模型

TPC-C 测试系统数据库由 9 张表组成，它们之间的关系如图 19-2 所示。

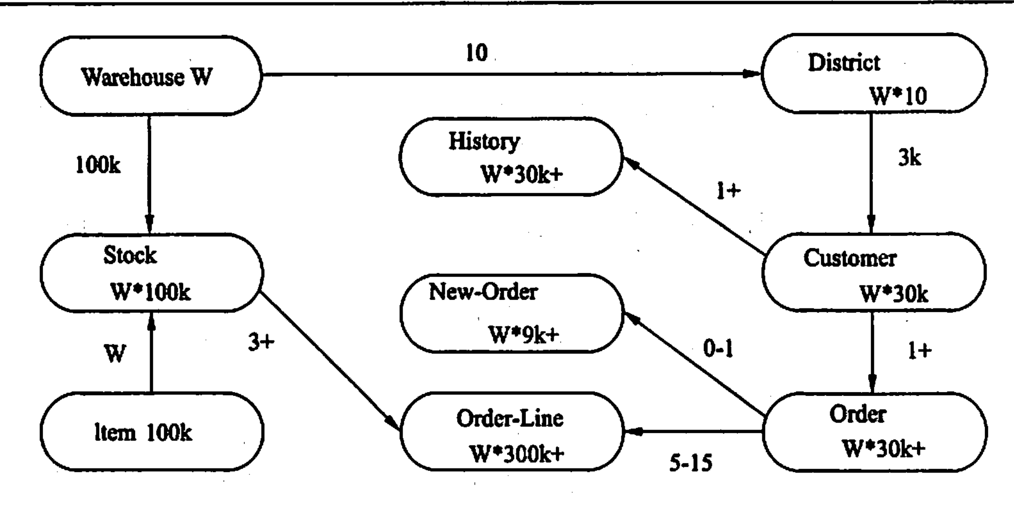


图 19-2　TPC-C 测试系统数据库

其中，表框里的数字表示该表将要存放多少条记录，仓库数 W 的调整在测试中能够体现数据库所能够支持的数据规模的能力；表间的数字表示表数据的父子关系之间儿子的个数，比如一个 Warehouse 要对应 10 个 District 等，另外，“+”号表示这种对应关系可能会更多。

3．事务说明

该系统需要处理的交易事务主要为以下几种。

- 新订单：客户输入一笔新的订货交易；
- 支付操作：更新客户账户余额以反映其支付状况；
- 发货：发货（模拟批处理交易）；
- 订单状态查询：查询客户最近交易的状态；
- 库存状态查询：查询仓库库存状况，以便能够及时补货。

有关事物的具体描述如下。

- 新订单：其主要事务内容为对于任意一个客户端，从固定的仓库中随机选取 5～15 件商品，创建新订单。其中 1%的订单，要由于假想的用户操作失败而回滚；该事务的主要特点为读写、频繁、要求响应快，是系统中最典型的操作，也是系统处理中的主要工作量，最终也是以数据库系统每分钟能够处理的新订单数来对数据库系统的性能进行评价。
- 支付操作：其主要事务内容为对于任意一个客户端，从固定仓库随机选取一个辖区及其内的用户，采用随机的金额支付一笔订单，并同时将该订单记录为相应历史订单。该事物的主要特点为 10 个批量、读写、较少、较宽松的响应时间。

- 订单状态查询：其主要事务内容为对于任意一个客户端，从固定仓库随机选取一个辖区及其内用户，读取该用户的最后一条订单，显示订单内每件商品的状态。该事物的主要特点为只读、较少、要求响应快。
- 发货：其主要事务内容为对于任意一个客户端，随机选取一个发货包，更新被处理订单的用户账户余额，并把修改后的订单从新订单中删除。该事物的主要特点为读写、频繁、响应快。
- 库存状态查询：其主要事务内容为对于任意一个客户端，从固定的仓库和辖区选取最后的20条订单，检查订单中所有货物的库存。计算并显示所有库存低于随机生成阈值的商品数量。该事物的主要特点为只读、较少、较为宽松的响应时间。

对于以上这5种类型的事务交易，前4种类型的交易要求响应时间在5秒钟以内；对于库存状况查询交易，要求响应时间在20秒以内。同时，这5种交易最终的比例还必须满足一定的要求，即支付操作的比例不得少于43%，订单状态查询、发货和库存状态查询的比例分别均不得少于4%。具体而言，5种事务要满足的时间、比例及隔离级别要求如表19-2所示。

表19-2 5种类型的事务交易

事务类型	事务最小百分比	最小键盘输入时间（秒）	90%事务响应时间要求（秒）	最小平均思考时间分布（秒）
新订单	n/a	18	5	12
支付操作	43.0	3	5	12
订单状态查询	4.0	2	5	10
发货	4.0	2	5	5
库存状态查询	4.0	2	20	5

4．测试指标

TPC-C测试规范经过两年的研制，于1992年7月发布。几乎所有在OLTP市场提供软硬件平台的国外主流厂商都发布了相应的TPC-C测试结果，随着计算机技术的不断发展，这些测试结果也在不断刷新。

TPC-C测试的结果主要有两个指标，即流量指标（Throughput，简称tpmC）和性价比（Price/Performance，简称Price/tpmC）。

- 流量指标（Throughput，简称tpmC）：按照TPC组织的定义，流量指标描述了系统在执行支付操作、订单状态查询、发货和库存状态查询这4种交易的同时，每分钟可以处理多少个新订单交易。所有交易的响应时间必须满足TPC-C测试规范的要求，且各种交易数量所占的比例也应该满足TPC-C测试规范的要求。在这种情况下，流量指标值越大说明系统的联机事务处理能力越高。

- 性价比（Price/Performance，简称 Price/tpmC）：即测试系统的整体价格与流量指标的比值，在获得相同的 tpmC 值的情况下，价格越低越好。

5．测试工具

按照 TPC-C 测试规范要求，测试工具和模型可以由厂商自行实现。在本次测试中，我们按照 TPC-C 标准规范自行开发了相应的测试工具。该工具采用 SEDA 型架构，可适用于高并发度的服务器程序，采用此架构能够提高客户端的并发能力，满足 TPC-C 测试要求的服务器负荷要求。

6．测试流程

tpmC 的最终值是在满足 TPC-C 标准规范要求的响应延迟下，系统可能达到的最大处理速度。所以，测试流程事实上是一个反复逼近的过程，即不断地寻找系统可能达到的最大值的过程，一般而言，可以先预估一个大概的数据装载量，然后在此条件下进行 TPC-C 测试，如果系统新订单及其他事务的响应时间均很好地满足要求，则可以加大数据量再进行测试，直到系统响应时间不能满足要求为止，在满足要求的情况下的 tpmC 值就是需要的最大 tpmC 值。具体流程如图 19-3 所示。

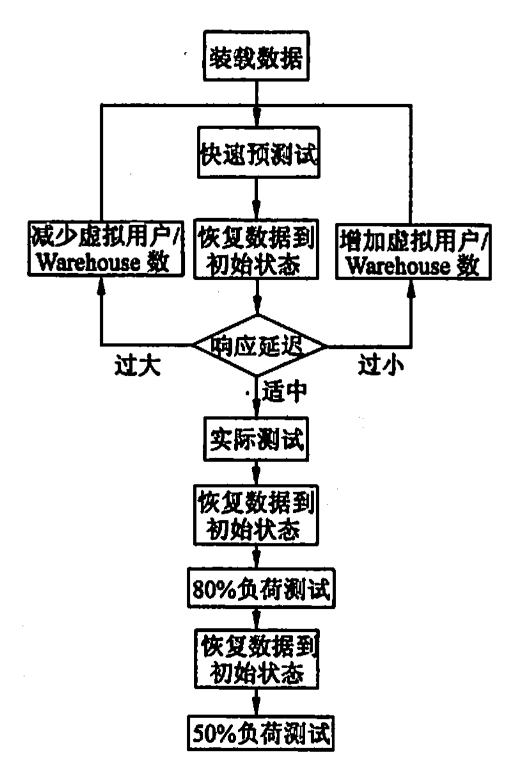


图 19-3 测试流程图



7．测试结果

根据 TPC 组织的规定，各厂商的 TPC-C 测试结果都按两种形式发布：测试结果概要（Executive Summary）和详细测试报告（Full Disclosure Report）。测试结果概要中描述了主要的测试指标、测试环境示意图以及完整的系统配置与报价，而详细测试报告中除了包含上述内容外，还详细说明了整个测试环境的设置与测试过程。

我们的测试报告基本按以上要求实现，但忽略了价格因素（由于目的不一样，我们的测试环境是统一的）。在测试报告中，我们包括的主要内容有：确认的 tpmC 峰值及其度量参数（包括 5 种事务的响应时间、事务所占的比例、测试时间等）、新订单响应时间分布曲线、每分钟新订单事务个数统计图以及详细的测试过程及配置说明等。

19.4.3 TPC-W 测试

1．测试概要

TPC-W 测试规范是专门针对电子商务应用系统的性能测试规范。该类系统具有的比较鲜明的特点如下。

- 复杂的在线浏览会话；
- 动态生成页面实现对数据库的访问和更新；
- 相对固定的页面对象；
- 模拟复杂的并发访问事物处理；
- 在线事物处理模式；
- 数据库由许多大小不一、属性多样，而又相互关联的数据表组成；
- 事务的完整性要求（ACID）；
- 存在较多数据访问和更新之间的资源争夺。

TPC-W 测试规范可以用来评测电子商务环境下服务器的处理性能。它模拟一个网上零售书店，通过远程浏览仿真器（Remote Browser Emulator，RBE）来模拟用户在网上浏览并订购相应的书籍。库存的书籍数量是 TPC-W 测试规范的一个扩展因子（Scale Factor），可以是 1000、10 000、100 000、1 000 000、10 000 000，在比较测试结果时，不能把不同扩展因子下的测试结果进行相互比较。

2．测试模型

TPC-W 测试环境需要建立一个网站，模拟用户登录此网站后可以浏览书籍分类、新到书籍、畅销书籍等相关栏目，同时系统还提供了按各种条件的查询功能，以及通过购物车订购书籍的功能。

该测试模型包含 8 张表，这些表的相互关系如图 19-4 所示。

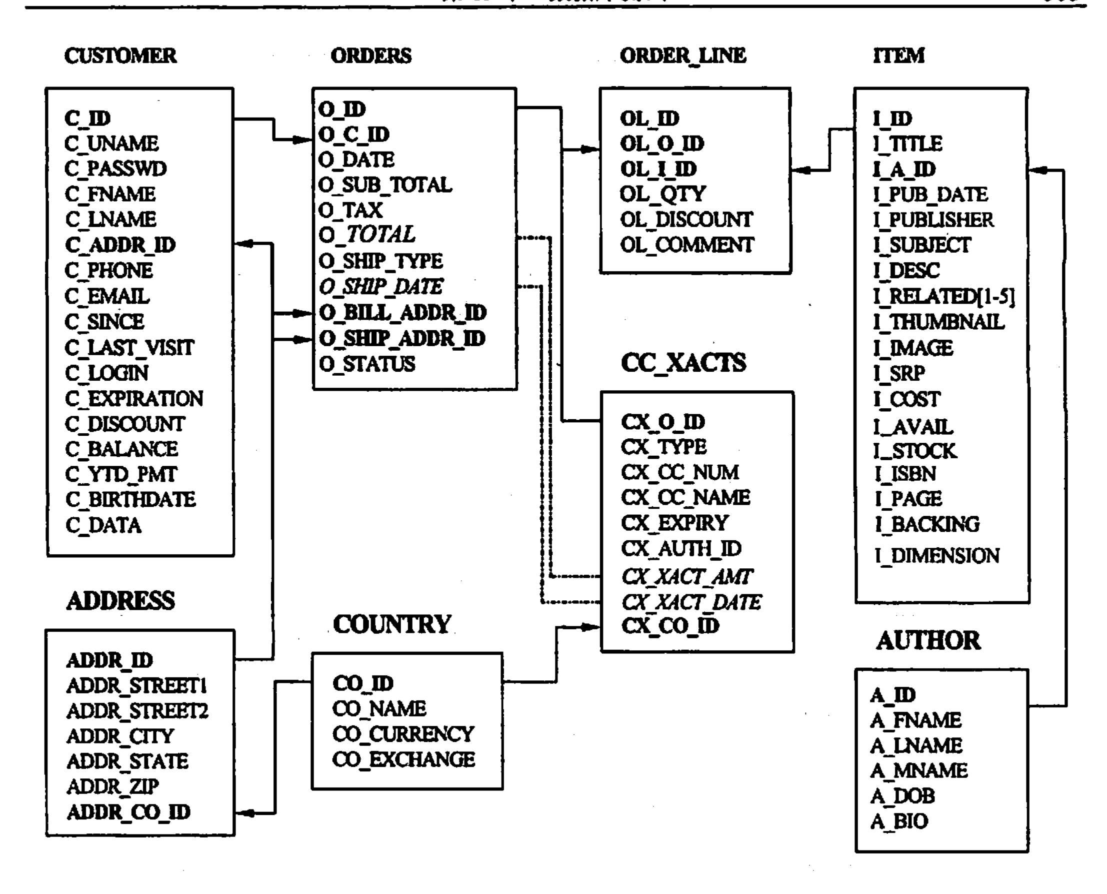


图 19-4　TPC-W 模型数据表

这 8 张表应该满足的数量关系如表 19-3 所示。

表 19-3　TPC-W 模型数据关系

Table Name	Cardinality (in rows)	Typical Row Length (in bytes)	Typical Table Size (in bytes)
CUSTOMER	2880 * (number of EB)	760	2,188,888 k
COUNTRY	92	70	6.44 k
ADDRESS	2 * CUSTOMER	154	887,040 k
ORDERS	0.9 * CUSTOMER	220	570,240 k
ORDER_LINE	3 * ORDERS	132	1,026,432 k
AUTHOR	0.25 * ITEM	630	1,575 k
CC_XACTS	1 * ORDERS	80	207,360 k
ITEM	1k, 10k, 100k, 1M, 10M	860	8,600 k

其中，表大小给出的是在 1000 个 EB（Enterprise Brower）和 10000 个商品的情况下的参考值；商品表（ITEM 表）的大小不包含图像的大小；除订单明细表（ORDER_LINE 表）的数据量可以根据订单表（ORDER 表）的数据量有所微调外，其余表的数据量不得改变，即必须满足以上要求。

3. 事物说明

根据 TPC-W 标准规范要求，该系统需要处理的交易事务主要为 14 类，即系统应该提供 14 个 Web 页面与终端用户进行交互，每个虚拟用户在选择一个页面后，可以随机地进入到另外一个页面。虚拟用户在浏览页面时可以有若干秒的思考时间，且平均的思考时间在 7～8 秒之间。

TPC-W 规定的 14 个进行交互操作的 Web 页面的响应时间要求如表 19-4 所示，同时，根据 TPC-W 规定，在整个测试周期内，每个页面交互的平均响应时间长于其对应的页面交互的 90%的响应时间不得超过 1 秒钟。

表 19-4 响应时间表

Web Interactions	90% WIRT Constraint/（秒）
Admin Confirm	20
Admin Request	3
Best Sellers	5
Buy Confirm	5
Buy Request	3
Customer Regist.	3
Home	3
New Products	5
Order Display	3
Order Inquiry	3
Product Detail	3
Search Request	3
Search Results	10
Shopping Cart	3

根据以上 14 种页面交互的特征，我们可以大致把以上 14 种页面交互操作分成两大类，即页面浏览类和订单订购类。由于在实际业务中，有时用户主要进行的是页面浏览操作，而提交订单的操作较少，有时用户主要是在进行订单提交而没有过多的浏览操作，还有一种相对比较均衡的时候，即浏览页面和提交订单比较均匀，属于正常购物。TPC-W 对此三种不同的业务模式都进行了定义：偏重浏览类（WIPSb）、偏重订单类（WIPSo）和正常购物类（WIPS），并针对这三种不同的业务模式，分别定义了它们各自包含的 14

个交互操作所占事务的比例，具体如表 19-5 所示。

表 19-5　页面交互混合比例

Web Interaction	Browsing Mix（WIPSb）（%）	Shopping Mix（WIPS）（%）	Ordering Mix（WIPSo）（%）
Browse	**95**	**80**	**50**
Home	29.00	16.00	9.12
New Products	11.00	5.00	0.46
Best Sellers	11.00	5.00	0.46
Product Detail	21.00	17.00	12.35
Search Request	12.00	20.00	14.53
Search Results	11.00	17.00	13.08
Order	**5**	**20**	**50**
Shopping Cart	2.00	11.60	13.53
Customer Registration	0.82	3.00	12.86
Buy Request	0.75	2.60	12.73
Buy Confirm	0.69	1.20	10.18
Order Inquiry	0.30	0.75	0.25
Order Display	0.25	0.66	0.22
Admin Request	0.10	0.10	0.12
Admin Confirm	0.09	0.09	0.11

4．测试指标

TPC-W 测试规范于 1999 年 12 月发布，其测试结果主要有两个指标，即性能指标（以 WIPS@scale factor 来进行度量，以下简称 WIPS）和性价比（简称$/WIPS），同时为更好地体现不同侧重点的交易情况，TPC-W 标准规范还定义了两个次要的辅助性能指标，即 WIPSb@scale factor（简称 WIPSb）和 WIPSo@scale factor（简称 WIPSo）。

- 性能指标（TPC-W Performance）：TPC-W 性能指标是用每秒钟可以进行 Web 交互的数量来衡量的，称为 WIPS（Web Interactions Per Second）。所有页面交互的响应时间必须满足 TPC-W 标准规范的要求，且各种页面交互数量所占的比例也应该满足 TPC-W 标准规范的要求。在这种情况下，性能指标值越大，说明系统的事务处理能力越高。同时，有关 WIPSb 和 WIPSo 的测试指标也应该包括在测试报告中，它们的测试方法与 WIPS 相同，但包含的 14 种事务比例各不相同。
- 性价比（简称$/WIPS）：即测试系统价格与性能指标的比值，在获得相同的 WIPS 值的情况下，价格越低越好。

5．测试工具

按照 TPC-W 测试规范要求，测试工具和模型可以由厂商自行实现。在本次测试中，

我们按照 TPC-W 标准规范，在参照 wisconsin 开发的测试工具的基础上进行大量修改，形成自己的测试工具。该测试工具与数据库管理系统采用 JDBC 连接，客户端、Web 服务器与数据库服务器分别部署。且 Web 服务器采用 Java Servlet 技术，图片对象采用文件系统存储实现。本次测试中库存的书籍数量为 10 000 种书籍。

6. 测试过程

TPC-W 测试过程就是寻找被测数据库管理系统在满足 TPC-W 标准规范要求的条件下的最大 WIPS 值、最大 WIPSb 值和最大 WIPSo 值的过程。由于 TPC-W 测试的主要度量指标为 WIPS，所以首先应该寻找到最大的 WIPS 值，然后在此基础上，寻找最大的 WIPSb 值和最大的 WIPSo 值。在这三种不同业务模式的测试过程中，系统软硬件环境必须一致，但仿真浏览器的数量及数据库的优化配置，如索引方案、Web 服务器的配置参数等可以不同。具体的测试流程示意图如图 19-5 所示。

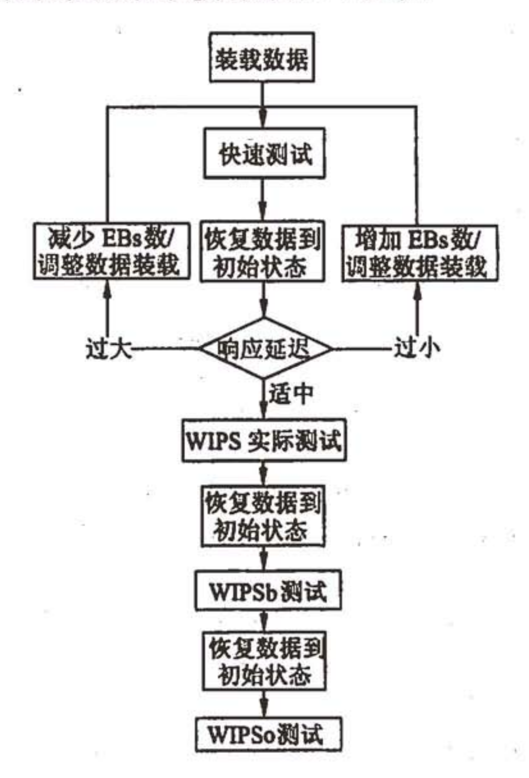


图 19-5 TPC-W 测试过程图

7. 结果分析

根据 TPC 组织的规定，各厂商的 TPC-W 测试结果都按两种形式发布：测试结果概

要（Executive Summary）和详细测试报告（Full Disclosure Report）。测试结果概要中描述了主要的测试指标、测试环境示意图以及完整的系统配置与报价，而详细测试报告中除了包含上述内容外，还详细说明了整个测试环境的设置与测试过程。

我们的测试报告基本按以上的要求实现，但忽略了价格因素（由于目的不一样，我们的测试环境是统一的）。在测试报告中，我们包括的主要内容有：性能指标值、WIPS 的交互比例统计、每秒钟 WIPS 值统计图、WIPSo 的交互比例统计、每秒钟 WIPSo 值统计图、WIPSb 的交互比例统计、每秒钟 WIPSb 值统计图、测试过程参数以及详细的测试过程及配置说明等。

19.4.4　解读 TPC 组织公布的性能测试报告

目前，交易处理委员会（TPC）发布的一系列测试规范都是人们评估数据库/仓库性能的主要参考标准，但如何去分析这些测试报告，如何根据这些测试报告正确地选择适合自己企业的数据库应用系统，依然是令许多用户感到困惑的问题。在本章最后，就此问题向大家作一个简单的介绍。

很多用户在使用 TPC 发布的测试结果时，只参考其测试概要中的系统性能指标和性价比指标，而忽略了对其整个测试环境、测试条件的完整分析。而事实上，各厂商从市场效应等各种因素出发，都会尽可能地对系统进行调整和优化，使得测试系统的性能达到最优状态，但这与实际生产环境往往有较大的差异。因此，建议用户在参考厂商提供的 TPC 指标时，除了研究其测试结果外，还应该仔细分析其详细测试报告，以便对其测试过程、优化手段等有一个全面的了解。

1．检查测试环境的建立过程

在详细测试报告中，商家都给出了建立测试环境时用到的所有程序脚本。不同平台的测试环境建立过程差别很大，有些只使用不到 3 页的程序脚本，而有些测试的程序脚本多达五六十页，而且很多参数的设定都依赖于数据库管理员的经验。同时，随着数据量的增加，系统测试环境建立的工作量和复杂度也可能发生变化。因此，我们有必要分析其建立测试环境的复杂程度，如果过于复杂，很显然，系统管理就会很复杂。

另外，我们还应该考虑测试系统是否具有高可靠性。很多测试系统都由多台服务器通过节点互联网络连成一个数据库环境，如果系统具有很高的可靠性，则当其中一台或多台服务器发生故障时，数据库系统仍能正常工作。很显然，这种高可靠性将影响测试系统在正常情况下的处理性能及性价比指标，因此大部分测试系统不具备高可靠性。这样的系统只能在实验室环境中作为测试用，不能拿到真正的生产环境下运行。

2．挤掉性价比指标的水份

在引用性价比指标时，需要注意提供测试系统价格的条件。有的厂商为了突出其性



价比指标，对测试系统提供较大的折扣，而获得这种折扣的条件只有在详细测试报告中才会加以说明。此外还需要注意的一点是，用于计算性价比指标的测试系统价格是各厂商在美国的报价，这与在中国的报价有很大的差别。

3．检查数据库的优化配置过程

各厂商为了取得好的测试性能指标，都会对系统进行调整和优化，而其中有些配置过程非常复杂或者需要的时间会很长，这些都反映了该系统的可管理性和可使用性。很显然，在实际应用中，我们不可能有这么长的时间和精力去等待和实现这些优化，至少它所需要的管理和维护成本太高。

4．比较测试的日期与可以供货的日期

在测试报告中，除了说明完成测试的日期外，还会说明系统可以供货的日期。一般来说，由于对系统进行各种优化以及技术的不断更新，后来进行测试的结果一般都会比以前的测试结果好很多。如果系统可供货日期比测试日期晚，说明测试时系统还局限于实验室研发，没有正式用户。

5．分析系统的可扩展性

正如前面所说，企业信息系统投入使用以后，采集的数据越来越多，同时企业的业务量也在不断增长，从而对信息系统的升级变得越来越迫切，因此数据库系统的可扩展能力也显得越来越重要。如果有些测试系统的配置本身已经到了极限，没有任何可以扩展的空间，在引用其结果时也应加以注意。

除此以外，我们在使用和参照这些结果时还应牢记以下两点：

其一，TPC 组织公布的性能测试结果中，一些产品的领先地位是暂时的，后来的测试结果一般比以前的好，这是因为测试模型是对外公布的，测试程序是厂商编写的，每个厂商都有足够的时间针对这些操作进行合理的优化。

其二，实验室中的性能测试往往不能反映现实情况，因为现实应用的环境和业务逻辑要复杂得多，在这些不同的业务逻辑面前，我们不能简单地将系统性能指标进行类比，所以，我们还应该针对自己的应用系统，设计专门的性能测试方案。

第20章　负载压力测试及故障定位与分析

20.1　测试需求分析

20.1.1　系统概述

《工程建设项目信息管理系统》是国家级项目，在全国范围内使用。使用的用户包括公司总部及其下属的部门及项目点。该系统主要完成与业务项目有关的管理工作，实现项目管理的流程化、系统化以及自动化。主要功能模块包括：结构化制度文档上传与下载、非结构化制度文档上传与下载、项目管理以及工作办公记事等。

系统的部署情况如下：Portal服务器和CM服务器设置在公司总部，总部下属的部门及项目点设置在全国各个省市，总部的连接方式采用100M局域网，部门及项目点与总部采用多种连接方式，包括宽带、ISDN、企业骨干网以及拨号等，系统软硬件以及网络环境见本章中测试环境的相关描述。

20.1.2　用户需求描述

该系统并发用户数为100个，模拟业务涉及下属部门和项目点，每个下属部门和项目点各有两个连接点，其中下属部门连接数为15×2（目前应用为10个下属部门），项目点接数为35×2（目前应用为23个项目点）。

业务类型主要是对文档的上传和下载，文档类型主要以图纸类非结构化数据为主，同时也存在少量结构化普通数据。

从外网访问该系统，某些操作速度很慢，希望找到故障瓶颈，以利于系统优化。

20.1.3　测试需求分析

1．需求分析概述

准备对该系统实施两次性能测试。第一次性能测试为性能检测与故障定位；第二次性能测试为对调优之后的效果进行评估。

第一次性能测试主要包括两个部分，局域网测试和广域网测试。

局域网测试主要内容为：在局域网测试环境下，对系统实施并发性能测试的同时，监控Portal服务器（IBM Websphere Portal 4.2、DB2 7.2、IBM AIX 5L ML 4）和CM服务



器（IBM Websphere 5.0、DB2 8.1、IBM AIX 5L ML 4）的资源使用情况。重点定位应用系统以及软、硬件支撑环境故障。

广域网测试主要内容为：在广域网环境下，对系统实施并发性能测试的同时，监控网络资源，同时关注Portal服务器及CM服务器资源使用情况。重点测试网络环境对应用系统的影响，定位网络故障，分析网络品质。

第二次性能测试针对第一次性能测试暴露的主要问题，测试并评估调优之后的系统性能。

测试指标及测试案例分析见后续章节。

2．测试指标分析原理与方法

（1）80～20 测试强度估算原理

每个工作日中80%的业务在20%的时间内完成。举例如下：每年业务量集中在8个月，每个月20个工作日，每个工作日8小时即每天80%的业务在1.6小时完成。

这里以案例“项目管理”中的检查点“条件查询”为例，来看80～20原理如何应用。

在过去的一年中，处理“条件查询”业务的约100万人次，其中15%的人对应用服务器提交7次查询请求；其中70%的人对应用服务器提交5次查询请求；其余15%的人对应用服务器提交3次查询请求。根据以往的统计结果，每年的业务增量为15%，考虑到今后3年业务发展的需要，测试需按现有业务量的两倍进行。

测试强度估算举例如下。

每年总的请求数为：（100×15%×7+100×70%×5+100×15%×3）×2=1000万次/年；

每天请求数为：1000/160=6.25万次/天；

每秒请求数为：（18750×80%）/（8×20%×3600）=8.68次/秒。

即服务器处理查询请求的能力应达到8.68次/秒。从后面的测试结果可以看到，系统的这项指标在并发用户数为50时达到3.56次/秒，此项指标值初步判为合格，但必须预测到当并发用户数达到预期值100时，该项指标可能会递增，需进一步考察系统性能。

其他指标的分析同理，用户可以自行计算。

（2）UCML 负载压力需求分析方法

根据需要可以采用UCML负载压力需求分析方法，具体使用方法可以参考“应用负载压力测试”章节。

3．测试案例分析要素

（1）任务分布情况

主要关注如下内容。

- 交易任务种类；
- 在一天的某些特定时刻系统都有哪些主要操作。

通过分析，可以得到例如下述有价值的信息：

- 在上午 10、11 点及下午 3、4 点，交易混合程度最高，混合比例如表 20-1 所示；
- 上午 9 点和下午两点是登录高峰期；
- 登录的并发用户数最大需支持 80，考虑业务扩充需求，最大并发用户数取 100；
- 工作记事的并发用户数最大需支持 50，考虑业务扩充需求，最大并发用户数取 80；
- 项目管理的并发用户数最大需支持 30，考虑业务扩充需求，最大并发用户数取 50；
- 退出的并发用户数最大需支持 80，考虑业务扩充需求，最大并发用户数取 100。

表 20-1　任务分布表

案例名称	并发用户数																							
登录									80	10	10			40	10	10	10							
制度文档-信息上传										20	20				20	20	10							
制度文档-文件上传与下载										20	20				20	20	10							
项目管理										20	20	20			30	20	30							
工作记事										10	20	20		10	20	20	50							
退出系统											10	60				10	80	10						
物理时刻	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

（2）交易混合情况

主要关注的内容包括：

- 系统日常业务主要有哪些操作，高峰期主要有哪些操作。
- 数据库操作有多少。
- 如果任务失败，那么商业风险有多少。

如表20-2所示为交易混合情况表。选择重点交易的指标为高吞吐量、高数据库I/O、高商业风险。对于本系统，“制度文档”、“项目管理”以及“工作记事”为负载压力测试重点。

（3）用户概况

主要关注的如下内容。

- 哪些任务是每个用户都要执行的。
- 针对每个用户组，不同任务的比例如何。



表 20-2 交易混合情况表

案例名称	日常业务	高峰期业务	Portal服务器负载	CM服务器负载	商业风险
登录	70/hr	210/hr	高	中	低
制度文档-信息上传	50/hr	160/hr	高	低	大
制度文档-文件上传与下载	50/hr	160/hr	中	高	大
项目管理	70/hr	200/hr	中	高	大
工作记事	100/hr	250/hr	高	中	中
退出系统	70/hr	210/hr	中	低	低

负载压力测试需要模拟不同部门用户的角色压力。如表20-3所示为用户概况表。

表 20-3 用户概况表

案例名称	文档室	项目室	上层领导
制度文档-信息上传	40		10
制度文档-文件上传与下载	40		10
项目管理		70	10
工作记事	20	30	70

4. 需求分析结论

以案例“项目管理”为例，在系统交易高峰期各项指标合格值如下：

- 系统并发用户数应该达到 50 个。
- 系统服务器处理请求的能力应达到 9 次/秒。
- 系统交易平均响应时间应在 10 秒之内。

20.2 测试案例制定

20.2.1 测试内容

1. 局域网测试

局域网测试的主要内容为：在局域网测试环境下，对系统实施并发性能测试的同时，监控Portal服务器（IBM Websphere Portal 4.2、DB2 7.2、IBM AIX 5L ML 4）和CM服务器（IBM Websphere 5.0、DB2 8.1、IBM AIX 5L ML 4）的资源使用情况。重点定位应用

系统以及软、硬件支撑环境故障。

（1）测试策略

- 对比测试环境。

① 对比测试环境 1（首页无频道，无 CM）；

② 对比测试环境 2（首页无频道，有 CM）。

注意，频道需要通过 CM 服务器加载，设置对比测试环境的主要目的是验证首页频道对系统性能的影响。

- 真实业务测试环境。

① 真实业务局域网测试环境（首页有频道，有 CM）；

② 真实业务机房测试环境（首页有频道，有 CM）。

注意，局域网测试环境模拟该系统局域网用户使用环境，机房测试环境力求排除网络因素，现实情况下，采用光纤连接负载生成器与服务器。

（2）测试用例

如表 20-4 所示为局域网测试案例。

表 20-4　局域网测试案例

案例名称		并发用户数（个）	网络环境(带宽)	数据量	备注说明
制度文档	信息上传	50、100	100M局域网 56Kbps Modem	50个用户并发，上传50条记录； 100个用户并发，上传100条记录	只上传信息， 不带附件
	文件上传下载	50、100		50个用户并发，上传50条记录； 100个用户并发，上传100条记录	信息和附件都上传 (附件大小为200k)
项目管理		50、100		50个用户并发，新增50条记录； 100个用户并发，新增100条记录	
工作记事		50、100		50个用户并发，新增50条记录； 100个用户并发，新增100条记录	

机房注：记录操作前后数据库记录数目；每个虚拟用户循环执行3次案例交易。

（3）测试范围

- 并发性能测试。
- 系统资源监控。

2．广域网测试

广域网测试的主要内容为：在广域网环境下，对系统实施并发性能测试的同时，监控网络资源，同时关注Portal服务器及CM服务器资源使用情况。重点测试网络环境对应用系统的影响，定位网络故障。

（1）测试策略



- 机房真实业务模拟环境；
- Internet 真实业务测试环境。

注意，机房真实业务模拟环境是指在机房环境进行测试，但利用网络分析工具模拟广域网真实业务；Internet真实业务测试环境指系统真实部署使用的环境。

（2）测试用例

如表 20-5 所示为广域网测试案例。

表 20-5 广域网测试案例

案例名称		并发用户数（个）	网络环境	数据量	备注说明
制度文档	信息上传	50、100	宽带（光纤）ISDN石化骨干网防火墙	50个用户并发，上传50条记录；100个用户并发，上传100条记录	只上传信息，不带附件
	文件上传下载	50、100		50个用户并发，上传50条记录；100个用户并发，上传100条记录	信息和附件都上传
项目管理		50、100		50个用户并发，新增50条记录；100个用户并发，新增100条记录	
工作记事		50、100		50个用户并发，新增50条记录；100个用户并发，新增100条记录	

注：记录操作前后数据库记录数目；每个虚拟用户循环执行3次。

（3）测试范围

- 并发性能测试；
- 系统资源监控；
- 系统应用在不同网络环境下的模拟；
- 系统应用在不同网络环境下的运行监控；
- 系统应用所在网络品质监控。

20.2.2 测试方法

采用自动化测试方法和人工测试方法相结合。

20.2.3 测试结果处理与分析

- 测试原始数据的统计与整理；

- 测试结果分析；
- 测试故障定位。

20.2.4 测试报告

在第一次性能测试结束时，出故障定位测试报告。第二次性能测试结束时，出性能测试与评估报告。

20.2.5 现场测试配合

现场测试人员如下。

- 中国软件评测中心测试工程师；
- 开发商技术支持；
- 测试软硬件环境技术支持；
- 用户代表。

20.3 测试环境、工具、数据准备

20.3.1 测试环境

1. 软、硬件配置

- Portal 服务器。

① 硬件。

机器型号：P670；

CPU：两个 powpc-power4 CPU，1.1GHz；

内存：3GB；

硬盘：两块 36GB 硬盘＋40GB 存储。

② 软件。

中间件：IBM Websphere Portal 4.2；

数据库：DB2 7.2；

操作系统：IBM AIX 5L ML 4。

- CM 服务器。

① 硬件。

机器型号：P670；

CPU：4 个 powpc-power4 CPU，1.1GHz；

内存：4GB；

硬盘：两块 36GB 硬盘。

② 软件。

中间件：IBM Websphere 5.0；

数据库：DB2 8.1；

操作系统：IBM AIX 5L ML 4；

内容管理器：IBM CM 8.1。

2．广域网环境网络拓扑图

如图 20-1 所示为广域网测试网络拓扑图。

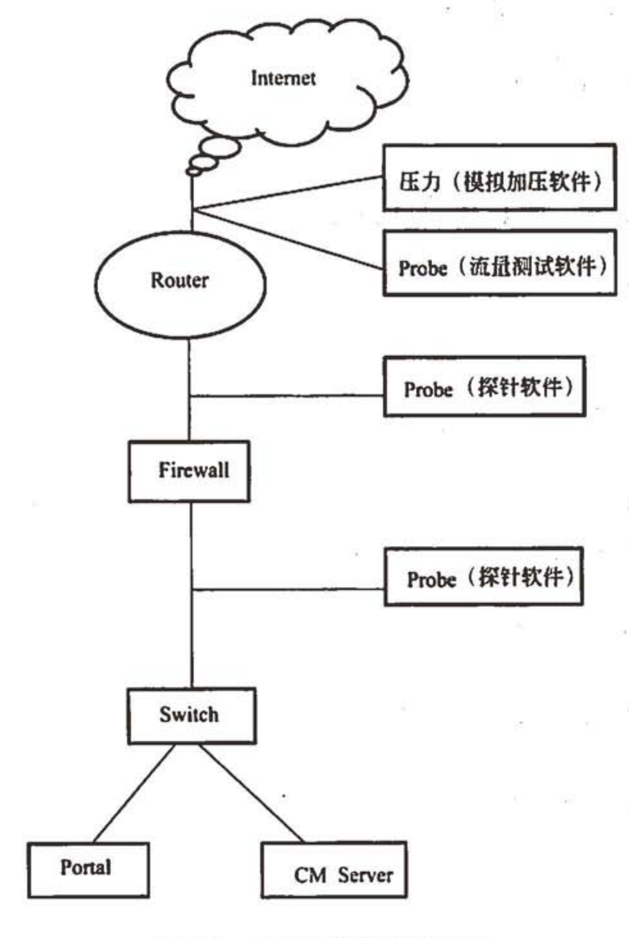


图 20-1 广域网测试网络拓扑图

3．测试用机

- 测试用机 1。

IBM T41 笔记本电脑。

CPU：Intel centrino 1.6GHz；

内存：512MB；

硬盘：40GB；

操作系统：Windows XP Professional 中文版；

工具软件：LoadRunner 7.8。

- 测试用机 2。

伦飞 P14 笔记本电脑。

CPU：Intel Pentium IV-M 2GHz；

内存：256MB；

硬盘：40GB；

操作系统：Windows 2000 Server 中文版；

工具软件：LoadRunner 7.8、Compuware Network Vantage 9.1、Compuware Application Vantage 9.1。

- 测试用机 3。

联想昭阳 E255 笔记本电脑。

CPU：Intel centrino 1.4GHz；

内存：256MB；

硬盘：40GB；

操作系统：Windows 2000 Professional 中文版；

工具软件：Mercury Interactive LoadRunner 7.8、Compuware Network Vantage 9.1、Compuware Application Vantage 9.1、Compuware Application Expert 9.1、Compuware Server Vantage 9.1。

20.3.2 测试工具

- LoadRunner V7.5.1（美国 Mercury Interactive 公司开发），用于并发性能测试；
- ServerVantage9.1（美国 Compuware 公司开发），用于代码级资源占用监控；
- NetworkVantage9.1（美国 Compuware 公司开发），用于网络监控；
- ApplicationVantage9.1（美国 Compuware 公司开发），用于网络应用故障定位；
- ApplicationExpert9.1（美国 Compuware 公司开发），用于应用在不同网络环境下的模拟。



20.3.3 测试数据

1．初始数据

初始数据需保证案例中业务操作正确完成。

2．业务数据

查询业务操作记录的数据准备，包括当前数据量，业务扩展预计数据量。例如：

- 案例“制度文档”中，附件上传文件需要模拟不同规模大小的文件；
- 案例“项目管理”中，数据库中项目记录数目等。

3．脚本中参数数据

数据池内容如下。

- 不同登录用户名与密码；
- “制度文档文件上传下载”所需不同种类的文件；
- 不同选择的查询条件。

20.4 测试脚本录制、编写与调试

脚本包括内容如下。

- 制度文档-信息上传；
- 制度文档-文件上传下载；
- 项目管理；
- 工作记事。

注意，每个脚本中都包括了登录与退出系统。

脚本的确定按照以下步骤进行。

① 脚本录制；

② 布置检查点；

③ 查找动态数据；

④ 参数替换；

⑤ 单脚本回放；

⑥ 脚本加压回放。

注意，步骤③中，如果有动态数据则编辑脚本，处理动态数据；若无动态数据，则可省略该步骤。

20.5 负载压力场景制定

主要包括以下内容。

- 每个虚拟用户 Action 的循环次数，定为 3。
- 每个虚拟用户 Action 的循环之间的间隔，定为无。
- 日志处理，定为有错误发生时传递消息，标准日志方式。
- 用户操作时间处理，定为忽略用户操作时间。
- 错误处理，错误不导致回放停止。
- 回放模式，定为线程方式。
- 交易处理，定为 Action 独立交易。
- 网络传输速度选择，定为最大带宽。
- 浏览器模拟方式，定为 IE6.0；模拟浏览器 cache；下载非 HTML 资源；在每个循环中模拟一个新用户。
- 代理处理，定为无。
- 脚本分发模式，定为无。
- 虚拟用户数，按照案例描述加载。
- 超时处理，设为 600s，注意，工具不同，选项处的设置保持一致。
- 回放内容检查，定为查询检查成功标志。
- 回放日程模式，定为场景模式；所有虚拟用户同时起停；交易完成时停止回放。
- 页面分解选项启动。
- 资源监控 counter 选择，定为 Portal 服务器（IBM Websphere Portal 4.2、DB2 7.2、IBM AIX 5L ML 4）和 CM 服务器（IBM Websphere 5.0、DB2 8.1、IBM AIX 5L ML 4），具体指标见测试结果。
- 其他项选择默认方式。

20.6 测试执行

20.6.1 执行成功标志

- 系统运行正常。
- 所有交易完全成功实现，并可重复执行 3 次，保证每次结果误差在可接收范围之内。

- 资源监控指标能够获取有效值。

20.6.2 实时监控内容

- 压力瓶颈报错信息。
- 交易执行关键指标。
- 资源监控关键指标。

交易成功执行之后，务必保存测试原始数据结果。

20.7 测试结果及分析

20.7.1 测试结果

1. 对比测试环境1和对比测试环境2

（1）交易执行情况

- 制度文档。如表 20-6 所示为制度文档交易执行情况。

表 20-6 制度文档交易执行情况

案例名称	并发用户数	交易响应时间(s)平均值			交易通过率(tran/s)最大值		
		查询所有	信息上传	文件下载	查询所有	信息上传	文件下载
信息上传	50 (f)	25.767	87.977		1.406	1.344	
文件上传下载(CM)	50 (f)	8.911	60.879	36.231	2.688	2.063	2.25

注：(f) 表示交易未能完全成功实现。

- 项目管理。如表 20-7 所示为项目管理交易执行情况。

表 20-7 项目管理交易执行情况

案例名称	并发用户数	交易响应时间(s)平均值			交易通过率(tran/s)最大值		
		项目选择	新增项目	条件查询	项目选择	新增项目	条件查询
项目管理	50 (f)	14.703	63.619	15.723	2.813	1.438	3.556
	50(CM)	23.804	68.511	11.471	1.813	1.313	2.5

注：(f) 表示交易未能完全成功实现。

- 工作记事。如表 20-8 所示为工作记事交易执行情况。

表 20-8　工作记事交易执行情况

案例名称	并发用户数	交易响应时间(s)平均值		交易通过率(tran/s)最大值	
		条件查询	新增记事	条件查询	新增记事
工作记事	50	4.777	6.411	4.875	3.5
	90	13.688	34.798	2	2.188
工作记事（CM）	90(f)	29.017	53.86	0.238	0.212

注：(f)表示交易未能完全成功实现。

（2）资源占用情况

- UNIX 资源。如表 20-9 所示为 UNIX 资源占用情况。

表 20-9　UNIX 资源占用情况

案例名称	并发用户数	CPU（%）		Page In/Page Out（num/s）		Disk Traffic（bytes/s）	
		CM 服务器	Portal 服务器	CM 服务器	Portal 服务器	CM 服务器	Portal 服务器
信息上传	50	0.666		0.015/0.087		1.724	
文件上传下载（CM）	50	3.874	23.487	0.087/12.952	0/0.259	7.594	0.519
项目管理	50	0.733	82.511	0/0.571	0.882/32.515	1.114	5.376
	50(CM)	0.597	88.504	0.022/1.233		2.35	
工作记事	50	0.398	69.427	0/0.482	0.213/7.046	0.933	5.462
	90	0.764	71.707	0/0.487	0.028/28.461	0.942	5.891
工作记事（CM）	90	0.885	71.096	0/0.6	0.006/15.822	1.167	2.934

- DB2 资源。如表 20-10 所示为 DB2 资源占用情况。

表 20-10　DB2 资源占用情况

监控指标	服务器	数据库	案例检查点（并发用户数）							
			信息上传	文件上传下载	项目管理		工作记事		工作记事（CM）	
			50	50	50	50(CM)	50	90	50	90
failed_sql_stmts（num）(L)	CM 服务器	RMDB	0	0	0		0	0		0
		ICMNLSDB	0	0	0		0	0		0
	Portal 服务器	ECIMSDB		5	152		2	1		0
locks_held（num）	CM 服务器	RMDB	0	0.11	0		0	0		0
		ICMNLSDB	0.112	4.321	0		0	0		0
	Portal 服务器	ECIMSDB		10.792	65.95		12.96	18.886		32.561



续表

监控指标	服务器	数据库	案例检查点（并发用户数）							
			信息上传	文件上传下载	项目管理		工作记事		工作记事（CM）	
			50	50	50	50(CM)	50	90	50	90
locks_waiting（num）	CM服务器	RMDB	0	0	0		0	0		0
		CMNLSDB	0	0	0		0	0		0
	Portal服务器	ECIMSDB		0.092	1.839		0.568	1.059		0
rollback_sql_stmts（num）(L)	CM服务器	RMDB	0	0	0		0	0		0
		CMNLSDB		5	152		2	1		0
	Portal服务器	ECIMSDB		153.833	62.206		0.864	4.395		157

注：(L)表示累计值。

- WebSphere 4.2 资源（Portal 服务器）。如表 20-11 所示为 WebSphere 4.2 资源（Portal 服务器）占用情况。

表 20-11 WebSphere 4.2 资源（Portal 服务器）占用情况

监控指标	案例检查点（并发用户数）							
	信息上传	文件上传下载	项目管理		工作记事		工作记事（CM）	
	50	50	50	50(CM)	50	90	50	90
AvgTimeWaitForLock（ms）		7286.007	6136.397		7314.304	8360.212		7750.652
NumWaitsForLock（num）		85635.785	68380.069		179791.783	227751.017		102244.143
freeMemory（byte）		154385315.494	150958556.23		150933229.867	141687711.586		148709897.016
totalMemory（byte）		308685467.544	306356665.379		268892672	290551190.069		305789440
Invalidated Sessions（num）		34	31.161		350.367	842.707		189.566
Concurrent Requests（num）		10.425	6.959		4.536	9.033		9.84
NumErrors（num）		0	0		0	0		0
Response Time（ms）		1860.264	676.156		1086.277	1509.826		1255.514
totalRequest		29863.835	14753.103		54503.6	98380.793		50685.778

2. 真实业务局域网测试环境

(1) 交易执行情况

- 制度文档。制度文档交易执行情况如表 20-12 和表 20-13 所示。

表 20-12　制度文档交易执行情况 I

案例名称	并发用户数	交易响应时间(s)平均值				交易通过率(tran/s)最大值		
		登录	查询所有	信息上传	文件下载	查询所有	信息上传	文件下载
信息上传	30	52.708	4.32	12.842		2.5	1.875	
	50(f)	73.002	5.671	20.592		4.375	4	
文件上传下载	50	76.949	7.45	32.376	10.312	3.625	2.875	2

表 20-13　制度文档交易执行情况 II

案例名称	并发用户数	交易响应时间(s)平均值			交易通过率(tran/s)最大值		
		查询所有	信息上传	文件下载	查询所有	信息上传	文件下载
信息上传	40 (x1f)	6.402	22.387		3	4.375	
文件上传下载	40 (x1f)	6.576	35.183	9.886	2.5	3.125	3.5
	50(f)	9.129	43.307	12.249	3.375	2.75	3.375

注：本组数据为发生虚拟用户不能循环执行案例交易情况之后的测试数据。

- 项目管理。项目管理交易执行情况如表 20-14 和表 20-15 所示。

表 20-14　项目管理交易执行情况 I

案例名称	并发用户数	交易响应时间(s)平均值				交易通过率(tran/s)最大值		
		登录	项目选择	新增项目	条件查询	项目选择	新增项目	条件查询
项目管理	50	142.66	27.094	71.921	12.361	1.188	1.063	1.5

表 20-15　项目管理交易执行情况 II

案例名称	并发用户数	交易响应时间(s)平均值			交易通过率(tran/s)最大值		
		项目选择	新增项目	条件查询	项目选择	新增项目	条件查询
项目管理	40 (x1f)	34.451	69.669	7.545	1.5	1.125	1.875
	50 (x1f)	46.552	86.105	10.992	2.75	1.875	2.333

注：本组数据为发生虚拟用户不能循环执行案例交易情况之后的测试数据。

- 工作记事。工作记事交易执行情况如表 20-16 和表 20-17 所示。

表 20-16　工作记事交易执行情况 I

案例名称	并发用户数	交易响应时间(s)平均值			交易通过率(tran/s)最大值	
		登录	条件查询	新增记事	条件查询	新增记事
工作记事	30	46.607	2.144	4.51	6	4.5



表 20-17 工作记事交易执行情况 II

案例名称	并发用户数	交易响应时间(s)平均值		交易通过率(tran/s)最大值	
		条件查询	新增记事	条件查询	新增记事
工作记事	30(f)	2.271	4.244	4.25	3.375
	30 (x1)	2.341	3.86	6.25	6.25
	50 (x1f)	2.557	4.933	4.625	5.333

注：本组数据为发生虚拟用户不能循环执行案例交易情况之后的测试数据。

- 混合案例。混合案例交易执行情况如表 20-18 所示。

表 20-18 混合案例交易执行情况

案例名称	并发用户数	交易响应时间（s）平均值				交易通过率（tran/s）最大值		
		登录	查询所有	信息上传	文件下载	查询所有	信息上传	文件下载
信息上传	10	168.615	8.815	19.198		0.625	0.625	
文件上传下载	10		9.839	21.692	6.882	1	1	0.875
项目管理	10		33.624	43.109	3.389	1.25	1	
工作记事	20		3.893	10.51	3	2	3.893	10.51

（2）资源占用情况

- UNIX 资源。如表 20-19 所示为 UNIX 资源占用情况。

表 20-19 UNIX 资源占用情况

案例名称	并发用户数	CPU（%）		Page In/Page Out（num/s）		Disk Traffic（bytes/s）	
		CM 服务器（登录）	Portal 服务器（最大值）	CM 服务器	Portal 服务器	CM 服务器	Portal 服务器
信息上传	30	19.862 100	65.161 100	0.654/11.778	0.008/5.763	23.021	5.692
	50(f)	18.072 59.739	74.142 100	0.053/6.14	0/1.59	3.347	4.965
文件上传下载	50	23.495 84.181	76.131 100	0.106/24.092	0.039/5.402	12.714	4.426

续表

案例名称	并发用户数	CPU（%）		Page In/Page Out（num/s）		Disk Traffic（bytes/s）	
		CM服务器（登录）	Portal服务器（最大值）	CM服务器	Portal服务器	CM服务器	Portal服务器
项目管理	50	17.281 68.158	84.511 100	0/3.397	0.267/74.587	4.735	5.99
工作记事	30	25.999 100	71.767 100	0/0.165	0.035/6.28	0.329	5.304

- DB2 资源。如表 20-20 所示为 DB2 资源占用情况。

表 20-20　DB2 资源占用情况

监控指标	服务器	数据库	案例检查点（并发用户数）				
			信息上传		文件上传下载	项目管理	工作记事
			30	50(f)	50	50	30
failed_sql_stmts（num）(L)	CM服务器	RMDB	0	0	0	0	0
		ICMNLSDB	0	0	0	0	0
	Portal服务器	ECIMSDB	7	8	3	118	0
locks_held（num）	CM服务器	RMDB	0	0	0.11	0	0
		ICMNLSDB	0	0	4.321	0	0
	Portal服务器	ECIMSDB	16.469	31.218	19.445	52.645	35.088
locks_waiting（num）	CM服务器	RMDB	0	0	0	0	0
		CMNLSDB	0	0	0	0	0
	Portal服务器	ECIMSDB	0	0	0	0.52	0
rollback_sql_stmts（num）(L)	CM服务器	RMDB	0	0	0	0	0
		CMNLSDB	0	0	0	0	0
	Portal服务器	ECIMSDB	7	8	3	118	0
commit_sql_stmts（L）	Portal服务器	ECIMSDB	838	1395	1790	1710	828
pool_data_l_reads	Portal服务器	ECIMSDB	0	0	0	0	0
pool_data_p_reads	Portal服务器	ECIMSDB	0	0	0	0	0
pool_data_writes	Portal服务器	ECIMSDB	0	0	0	0	0
pool_index_l_reads	Portal服务器	ECIMSDB	0	0	0	0	0



续表

监控指标	服务器	数据库	案例检查点（并发用户数）				
			信息上传		文件上传下载	项目管理	工作记事
			30	50(f)	50	50	30
pool_index_p_reads	Portal服务器	ECIMSDB	0	0	0	0	0
pool_index_writes	Portal服务器	ECIMSDB	0	0	0	0	0
dynamic_sql_stmts(L)	Portal服务器	ECIMSDB	12128	16402	20449	25368	9676
static_sql_stmts(L)	Portal服务器	ECIMSDB	845	1403	1793	1828	828
rows_deleted(L)	Portal服务器	ECIMSDB	0	0	0	0	0
rows_inserted(L)	Portal服务器	ECIMSDB	73	132	146	228	66
rows_selected(L)	Portal服务器	ECIMSDB	138113	212438	227194	241938	150104
rows_updated(L)	Portal服务器	ECIMSDB	80	140	299	346	66

注：(L)表示累计值。

- WebSphere 4.2 资源（Portal 服务器）。如表 20-21 所示为 WebSphere 4.2 资源（Portal 服务器）占用情况。

表 20-21 WebSphere 4.2 资源（Portal 服务器）占用情况

监控指标	案例检查点（并发用户数）				
	信息上传		文件上传下载	项目管理	工作记事
	30	50(f)	50	50	30
AvgTimeWaitForLock（ms）	8214.33	8052.68	8256.591	6202.77	7938.563
NumWaitsForLock（num）	289695.818	276549	301590.3	173875.521	267646.886
freeMemory（byte）	123039632.606	111167893.455	105373477.714	115881956.426	119752476.571
totalMemory（byte）	268892672	268982039.23	268892672	277703778.043	273283584
InvalidatedSessions（num）	749	697.8	795.429	75	589
ConcurrentRequests（num）	33.508	33.997	34.914	13.477	32.448
numErrors（num）	0	0	0	0	0
responseTime（ms）	7709.501	7826.233	7646.366	6771.615	7962.013
TotalRequest	92227.03	86995.2	100416.357	28078.266	79607.086

3. 真实业务机房测试环境

（1）交易执行情况

这里所列测试数据为“真实业务机房测试环境”与“真实业务局域网测试环境”对比数据。

- 制度文档。

① 信息上传（40×3、机房、未能成功实现；失败3个交易，进入业务系统）。制度文档交易执行情况如表20-22所示。

表 20-22　制度文档交易执行情况

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	32.827	40.216	69.064
vuser_end_Transaction	0	0	0
vuser_init_Transaction	85.503	150.832	204.778
新增信息上传	4.828	6.784	8.432
查询所有综合管理类	2.102	2.399	4.416
进入业务系统	7.939	21.436	24.977
进入系统管理	2.122	3.068	8.963

② 信息上传（40×3、局域网、成功实现）。如表20-23所示为信息上传交易执行情况。

表 20-23　信息上传交易执行情况

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	39.959	48.75	56.701
vuser_end_Transaction	0	0	0
vuser_init_Transaction	92.851	156.608	198.623
新增信息上传	7.04	9.077	12.027
查询所有综合管理类	2.484	3.57	6.279
进入业务系统	12.266	24.485	30.65
进入系统管理	2.433	5.038	12.137

③ 文件上传与下载（40×3、机房、成功实现）。如表20-24所示为执行情况Ⅰ。

表 20-24　文件上传与下载交易执行情况Ⅰ

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	37.474	66.105	91.091
vuser_end_Transaction	0	0	0
vuser_init_Transaction	83.027	139.966	188.003
文件下载	2.574	6.183	10.109
新增上传文件	5.899	11.704	16.644
查询全部集团	2.092	2.914	5.779
进入业务系统	11.182	32.925	50.169
进入系统管理	2.153	3.543	12.708



④ 文件上传与下载（40×3、局域网、成功实现）。如表20-25所示为执行情况Ⅱ。

表 20-25 文件上传与下载交易执行情况 II

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	51.404	67.578	81.357
vuser_end_Transaction	0	0	0
vuser_init_Transaction	38.369	140.266	204.877
文件下载	2.643	9.16	15.202
新增上传文件	10.064	17.735	26.939
查询全部集团	2.6	3.83	7.29
进入业务系统	12.305	23.854	27.908
进入系统管理	2.422	4.516	9.023

- 项目管理。

① 40×3、机房、未能成功实现（失败3个交易，两个进入业务系统，1个新增项目）。如表20-26所示为项目管理交易执行情况 I。

表 20-26 项目管理交易执行情况 I

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	98.067	122.883	146.489
vuser_end_Transaction	0	0	0
vuser_init_Transaction	112.786	162.547	209.339
新增项目	21.492	47.764	65.274
进入业务系统	12.331	21.35	29.543
项目查询	2.094	5.007	19.628
项目选择	11.04	28.153	39.647

② 40×3、局域网、成功实现。如表20-27所示为项目管理交易执行情况II。

表 20-27 项目管理交易执行情况 II

Transaction Name	Minimum	Average	Maximum
Action1_Transaction	102.546	154.709	232.435
vuser_end_Transaction	0	0	0
vuser_init_Transaction	134.453	184.903	229.132
新增项目	40.087	73.649	133.222
进入业务系统	15.626	27.21	56.521
项目查询	2.193	5.409	20.081
项目选择	10.494	37.495	87.922

- 工作记事。

① 30×3、机房、成功实现。其工作记事交易执行情况如表20-28所示。

表 20-28　工作记事交易执行情况 I

Transaction Name	Minimum	Average	Maximum
Action_Transaction	31.156	43.734	52.295
vuser_end_Transaction	0	0	0
vuser_init_Transaction	49.497	100.262	138.998
新增工作记事	3.345	4.861	8.991
条件查询	2.072	2.387	4.396
进入业务系统	8.863	16.119	21.559
进入工作记事	2.293	5.151	7.551
进入日常办公	1.001	1.001	1.009
选择日期	2.285	4.532	7.611

② 30×3、局域网、成功实现。其工作记事交易执行情况如表20-29所示。

表 20-29　工作记事交易执行情况 II

Transaction Name	Minimum	Average	Maximum
Action_Transaction	77.004	103.931	132.703
vuser_end_Transaction	0	0	0
vuser_init_Transaction	71.206	126.796	172.644
新增工作记事	6.659	21.088	37.916
条件查询	2.193	2.496	7.22
进入业务系统	15.22	21.074	25.165
进入工作记事	6.018	24.638	48.8
进入日常办公	1	1.002	1.01
选择日期	6.27	20.346	31.384

- 混合案例。

① 混合案例（20+10+10+10）×3、机房、成功实现。其执行情况如表20-30所示。

表 20-30　混合案例交易执行情况 I

Transaction Name	Minimum	Average	Maximum
Action_Transaction	34.921	176.576	322.464
Action1_Transaction	57.132	207.485	477.686
vuser_end_Transaction	0	0	0
vuser_init_Transaction	78.057	183.388	272.085
文件下载	3.584	18.491	29.042
新增上传文件	5.828	43.157	114.164
新增信息上传	4.957	35.219	106.162
新增工作记事	3.636	45.613	84.121

续表

Transaction Name	Minimum	Average	Maximum
新增项目	11.596	78.031	223.321
条件查询	2.102	5.882	9.847
查询全部集团	2.203	15.797	34.029
查询所有综合管理类	2.173	17.603	31.035
进入业务系统	5.337	57.936	150.246
进入业务系统	4.515	53.433	150.265
进入工作记事	2.844	34.102	50.072
进入日常办公	0.998	1.296	5.869
进入系统管理	2.223	25.514	58.053
选择日期	2.444	29.428	40.057
项目查询	2.153	10.159	22.022
项目选择	3.395	28.592	69.099

② 混合案例（20+10+10+10）×3、局域网、未能成功实现。其执行情况如表20-31所示。（失败15个交易，2个新增工作记事，9个进入业务系统，4个选择日期）

表 20-31 混合案例交易执行情况 II

Transaction Name	Minimum	Average	Maximum
Action_Transaction	64.99	96.177	112.407
Action1_Transaction	69.598	127.099	164.34
vuser_end_Transaction	0	0	0
vuser_init_Transaction	125.884	213.049	278.066
文件下载	37.289	54.491	65.78
新增上传文件	8.157	17.137	30.443
新增信息上传	5.045	13.701	24.926
新增工作记事	8.389	14.081	20.564
新增项目	22.501	56.424	84.575
条件查询	2.125	2.392	3.704
查询全部集团	2.319	7.304	15.569
查询所有综合管理类	2.441	7.845	16.511
进入业务系统	22.812	29.355	38.993
进入业务系统	12.969	26.31	39.296
进入工作记事	5.38	10.774	16.188
进入日常办公	1	1.001	1.016
进入系统管理	3.177	8.83	18.466
选择日期	3.428	12.49	22.489
项目查询	2.242	6.244	26.276
项目选择	9.245	22.233	43.32

注：混合案例比例设置同前；对于未完全成功的案例，上述指标统计的为成功交易的响应时间，这一点从某种程度上讲也降低了交易平均响应时间。

（2）资源占用情况

- UNIX 资源。如表 20-32 所示为 UNIX 资源占用的情况。

表 20-32　UNIX 资源占用情况

案例名称	并发用户数	CPU（%）		Page In/Page Out（num/s）		Disk Traffic（bytes/s）	
		CM服务器（登录）	Portal服务器（最大值）	CM服务器	Portal服务器	CM服务器	Portal服务器
信息上传	40(f)	58.223 100	80.734 100	0/2.553	4.444/56.665	5.009	5.61
文件上传下载	40	54.441 100	82.255 100	0.015/22.699	2.231/43.195	13.939	6.236
项目管理	40(f)	34.22 100	85.223 100	0/2.509	4.103/583.628	4.805	26.419
工作记事	30	43.403 100	75.188 100	0/3.225	9.3347/44.411	6.166	20.424
混合案例	共50	24.156 100	64.548 100	0.002/4.665	6.44/133.734	5.423	8.482

- DB2 资源：（Portal 服务器），其占用情况如表 20-33 所示。

表 20-33　DB2 资源（Portal 服务器）占用情况

监控指标	案例检查点（并发用户数）				
	信息上传	文件上传下载	项目管理	工作记事	混合案例
	40(f)	50	40(f)	30	50
failed_sql_stmts（num）(L)	4	4	13		3
locks_held（num）	35.809	33	150.107		20.732
locks_waiting（num）	0	0.731	0.454		0.018
rollback_sql_stmts（num）(L)	4	4	13		3
commit_sql_stmts（L）	1068	1548	1532		1610
pool_data_l_reads	0	0	0		0
pool_data_p_reads	0	0	0		0
pool_data_writes	0	0	0		0
pool_index_l_reads	0	0	0		0
pool_index_p_reads	0	0	0		0
pool_index_writes	0	0	0		0
dynamic_sql_stmts(L)	17629	22048	25326		35237



续表

监控指标	案例检查点（并发用户数）				
	信息上传	文件上传下载	项目管理	工作记事	混合案例
	40(f)	50	40(f)	30	50
static_sql_stmts(L)					
rows_deleted(L)	0	0	0		0
rows_inserted(L)	74	111	332		267
rows_selected(L)	235538	243289	288423		437934
rows_updated(L)	80	235	455		360

- WebSphere 4.2 资源（Portal 服务器），其占用情况如表 20-34 所示。

表 20-34　WebSphere 4.2 资源（Portal 服务器）占用情况

监控指标	案例检查点（并发用户数）				
	信息上传	文件上传下载	项目管理	工作记事	混合案例
	40(f)	40	40(f)	30	50
AvgTimeWaitForLock（ms）	9531.475	8813.093	9602.327	7508.557	9899.551
NumWaitsForLock（num）	220804.016	201662.731	230661.281	85779.056	2609333.328
freeMemory（byte）	104753042.625	122319162.746	91877367.281	95778101.185	107665960.826
TotalMemory（byte）	297859584	297859584	297859584	271065069.037	297859584
InvalidatedSessions（num）	612.442	611.463	678.618	5	1115.241
ConcurrentRequests（num）	120.118	105.298	126.197	7.341	134.689
numErrors（num）	0	0	0	0	0
ResponseTime（ms）	13130.036	13509.327	13063.327	4308.865	12788.056
totalRequest	96286.672	75048.06	106428.551	8018.981	128272.362

- 进程 CPU%与 Memory%明细（机房，Portal 服务器）：测试数据取测试周期内最大值作进程明细分析。
- 制度文档。

① 信息上传CPU%（40×3、未能成功实现）。制度文档CPU%明细分析如图20-2所示。

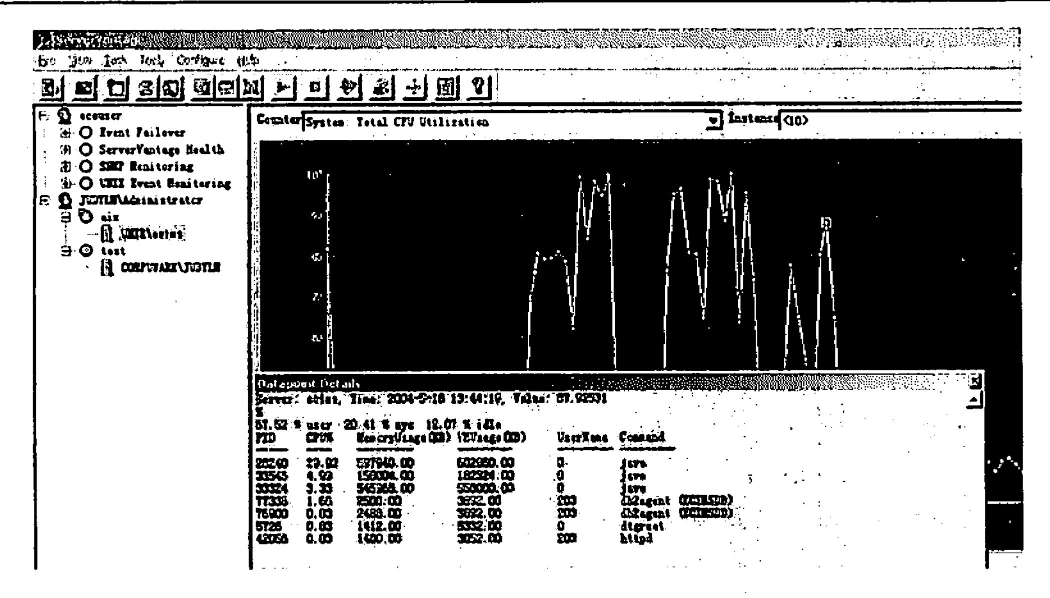


图 20-2　制度文档 CPU%明细分析

② 信息上传Memory%（40×3、未能成功实现）。制度文档Memory%明细分析如图20-3所示。

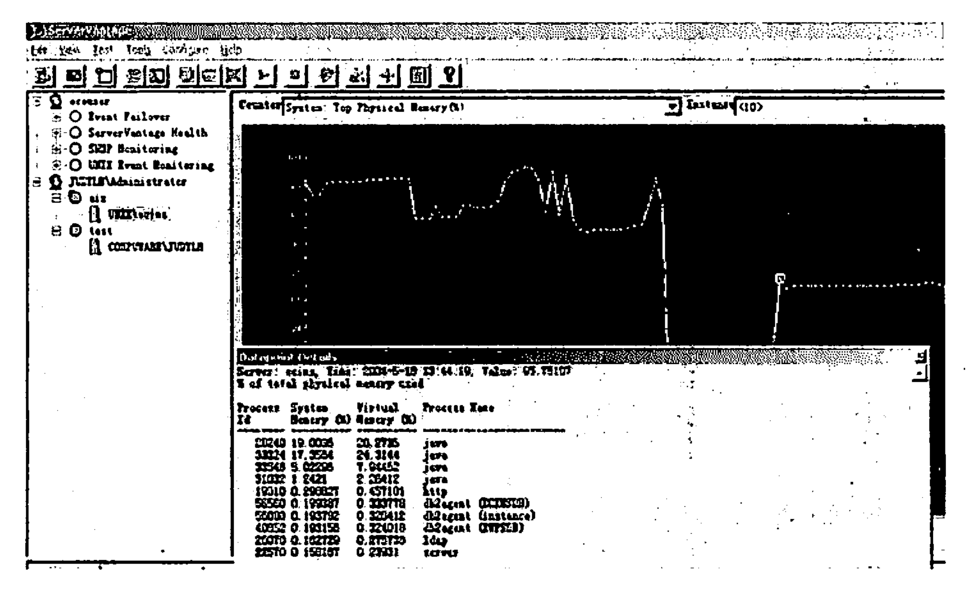


图 20-3　制度文档 Memory%明细分析

③ 文件上传与下载CPU%（40×3、成功实现）。其明细分析如图20-4所示。



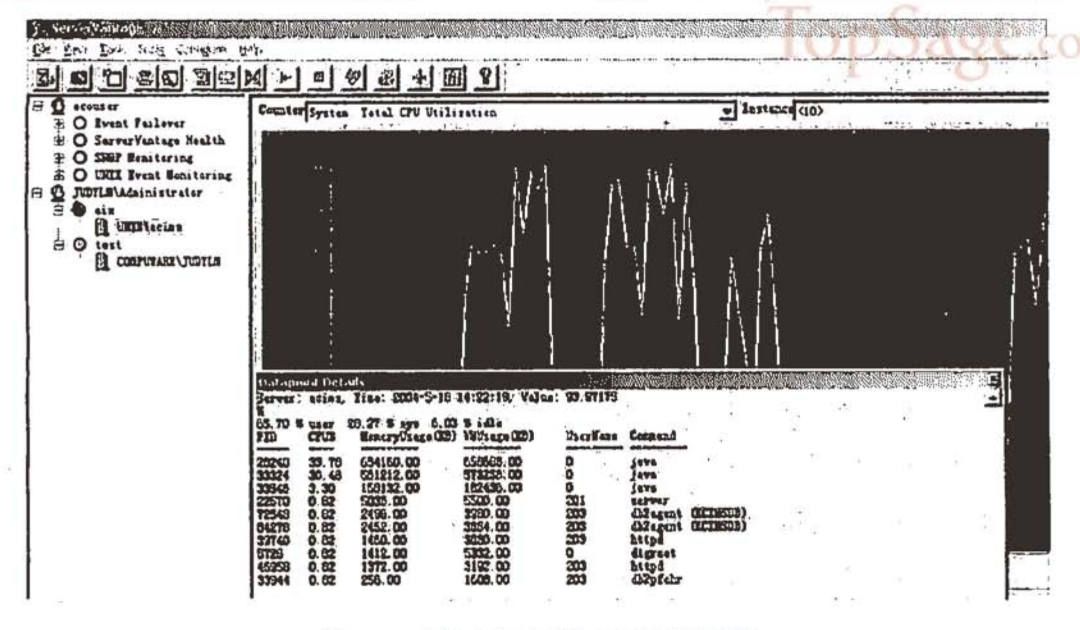


图 20-4 文件上传与下载 CPU%明细分析

④ 文件上传与下载Memory%（40×3、成功实现）。其明细分析如图20-5所示。

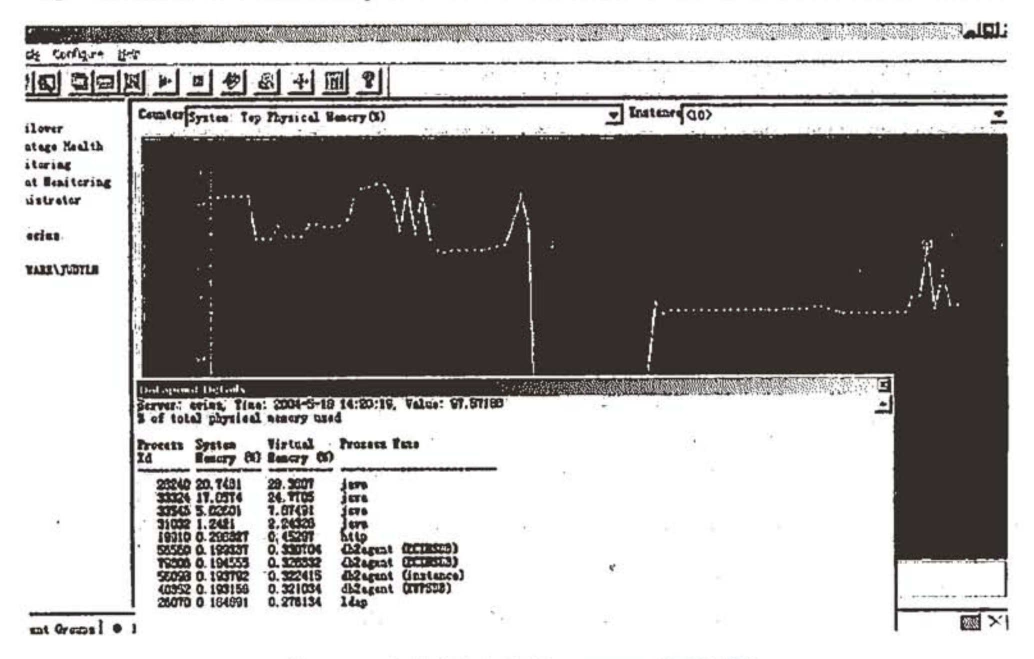


图 20-5 文件上传与下载 Memory%明细分析

- **项目管理：**

① CPU%、40×3、未能成功实现。其明细分析如图20-6所示。

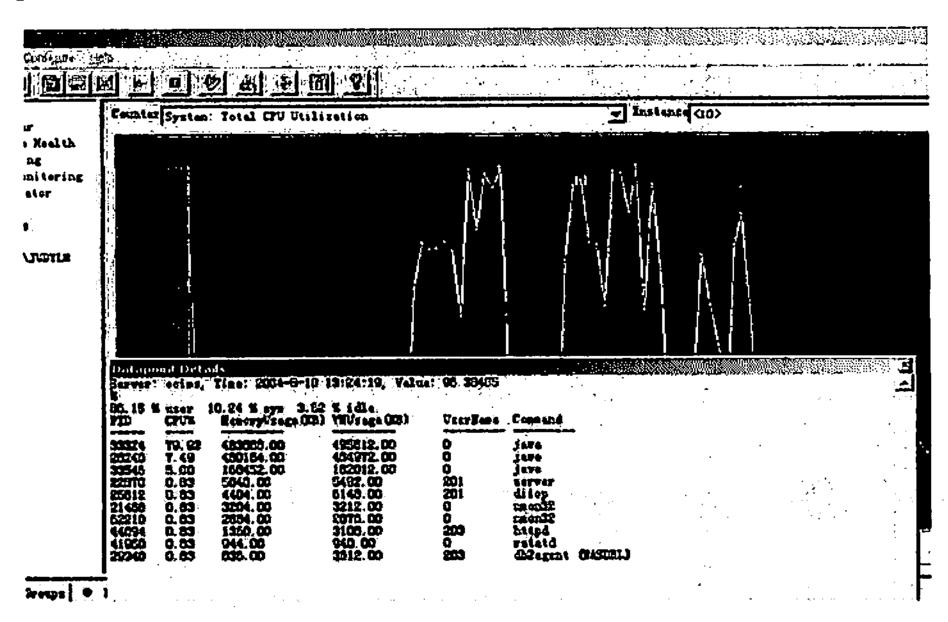


图 20-6　项目管理 CPU%明细分析

② Memory%、40×3、未能成功实现。其明细分析如图20-7所示。

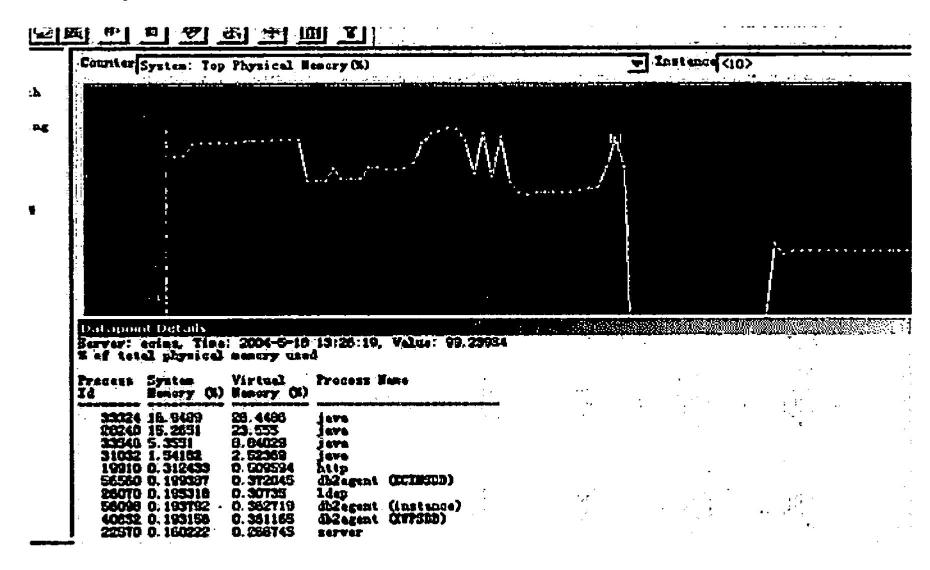
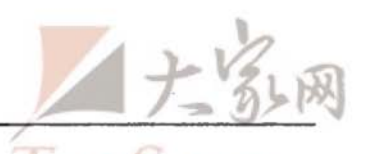


图 20-7　项目管理 Memory%明细分析



- 工作记事：

① CPU%、30×3、成功实现。其明细分析如图20-8所示。

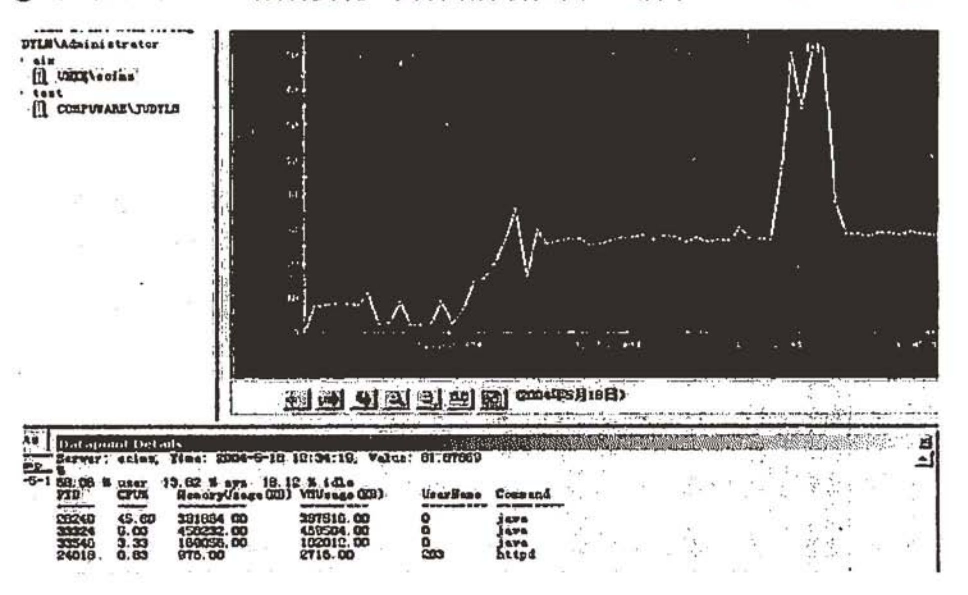


图 20-8　工作记事 CPU%明细分析

② Memory%、30×3、成功实现。其明细分析如图20-9所示。

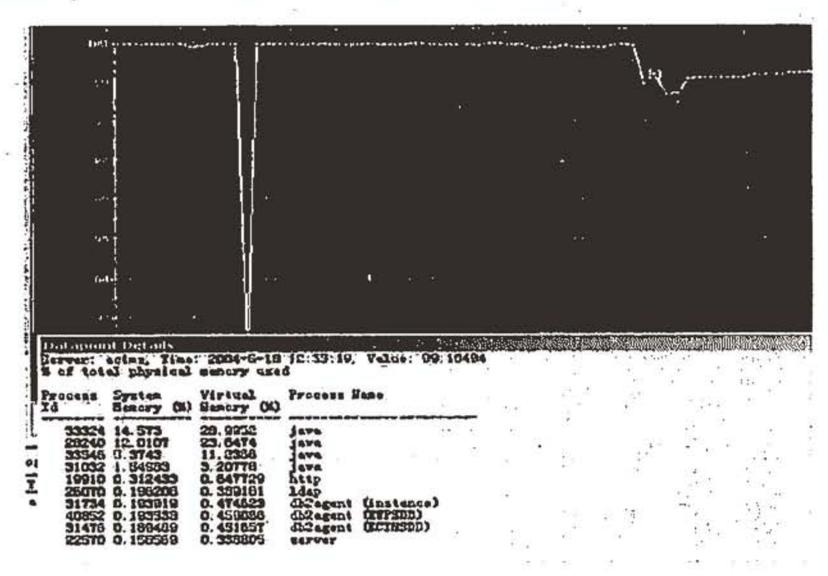


图 20-9　工作记事 Memory%明细分析

注：

- 不加说明指标均取平均值；
- (f)表示交易未能完全成功实现；
- (x1f)表示交易未能完全成功实现，循环执行 1 次；
- 其他监控指标值见测试原始数据（可以利用 Loadrunner 分析器查看，本书略）。

20.7.2　结果分析

1．局域网结果分析

（1）并发用户数

总的看来，下述错误主要集中在交易执行部分，登录部分很少报错。

- 案例“制度文档”，最大并发用户数未达到 50，

①“信息上传”报错：

Action1.c(57): Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON"（服务器响应错误）。

```
web_custom_request("×××.com.ausso.SSO_USER_LOGON",

"URL=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
        "Method=POST",
        "Resource=0",
        "RecContentType=text/html",

"Referer=http://ecims.××××.com.cn/wps/myPortal/.cmd/cs/.ce/155/.s/322/.r/1",
        "Snapshot=t6.inf",
        "Mode=HTTP",
"Body=userid=CSTC01&userpassword=CSTCSEI&servleturl=%2Fmain.jsp&reloginservleturl=%2Flogin.jsp&username=-&systemname=%E4%B8%AD%E7%9F%B3%E5%8C%96%E5%B7%A5%E7%A8%8B%E5%BB%BA%E8%AE%BE%E4%BF%A1%E6%81%AF%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F&sub_system_code=&sub_system_name=&sdet_lang=zh",
        LAST);
```

②“文件上传”报错：

Action1.c(3628): Error -27792: Failed to transmit data to network: [10053] Software caused connection abort 19（网络错误）。

```
web_custom_request("ECIMS.DT.BusinessProcess.DOC_STAN_UPLOAD_process",
```



```
"URL=http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.DOC_STAN_UPLOAD_process",
        "Method=POST",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://ecims.××××.com.cn/doc/doc_stan_input.jsp",
        "Snapshot=t17.inf",
        "Mode=HTTP",
        "Binary=1",
        "EncType=multipart/form-data;boundary=---------------------------7d49e3a5b0762",
        body1_ecims_dt_businessprocess_doc_stan_upload_process,
        LAST);
```

Action1.c(231): Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process" （服务器响应错误）。

```
web_custom_request("ECIMS.DT.BusinessProcess.COM_NEED_DO_process",

"URL=http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process",
        "Method=POST",
        "Resource=0",
        "RecContentType=text/html",
"Referer=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
        "Snapshot=t10.inf",
        "Mode=HTTP",
"Body=date=&where=+and+need_type+in+%28%27%CF%B5%CD%B3%D0%B4%C8%EB%27%2C
%27%B9%A4%D7%F7%C8%D5%B3%CC%27%29+and+status+%3D+%27%B4%FD%B4%A6%C0%ED
%27+and+substr%28alert_begin%2C1%2C10%29%3C%3D%272004-04-30%27+and+SUBSTR%28ALERT_
end%2C1%2C10%29%3E%3D%272004-04-30%27+and+EMP_ID+%3D%27CSTC01%27+order+by+ALE_
RT_BEGIN&act_type=select&userid=CSTC01&servleturl=%2Fhr%2Fcom_need_do_db_redirect_main.jsp&
StartRow=0&RowPerPage=30000",
        LAST);
```

- 案例“项目管理”，无 CM 最大并发用户数未达到 50，有 CM 最大并发用户数达到 50 “无 CM”报错：

vuser_init.c(7): Error -27796: Failed to connect to server "ecims.××××.com.cn:80": [10060] Connection timed out （服务器连接错误）。

```
web_url("ecims.××××.com.cn",
    "URL=http://ecims.××××.com.cn/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTTP",
    LAST);
```

- 案例“工作记事”，最大并发用户数达到 90，“并发用户数 100”报错：

Action1.c(60): Error -26612: HTTP Status-Code=500 (Internal Server Error) for http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON（服务器响应错误）。

```
web_submit_data("×××.com.ausso.SSO_USER_LOGON",
    "Action=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
    "Method=POST",
    "RecContentType=text/html",

"Referer=http://ecims.××××.com.cn/wps/myPortal/.cmd/cs/.ce/155/.s/322/.r/1",
    "Snapshot=t6.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=userid", "Value=CSTC01", ENDITEM,
    "Name=userpassword", "Value=CSTCSEI", ENDITEM,
    "Name=servleturl", "Value=/main.jsp", ENDITEM,
    "Name=reloginservleturl", "Value=/login.jsp", ENDITEM,
    "Name=username", "Value=-", ENDITEM,
    "Name=systemname", "Value=*************8?", ENDITEM,
    "Name=sub_system_code", "Value=", ENDITEM,
    "Name=sub_system_name", "Value=", ENDITEM,
    "Name=sdet_lang", "Value=zh", ENDITEM,
    LAST);
```

- 真实业务测试环境：案例“制度文档”，最大并发用户数未达到 50，“信息上传”报错如下。

Action1.c(270): Error -26627: HTTP Status-Code=404 (Not Found) for http://ecims.××××.com.cn/myPortal/.cmd/cs/.ce/155/.s/322/.r/1（请求的资源未找到）。



```
web_custom_request("j_security_check",
      "URL=http://ecims.××××.com.cn/j_security_check",
      "Method=POST",
      "Resource=0",
      "RecContentType=text/html",
      "Referer=http://ecims.××××.com.cn/ssoBody.jsp",
      "Mode=HTTP",
      "Body=j_username=CSTC01&j_password=CSTCSEI",
      LAST);
```

- 真实业务测试环境：案例“文件上传下载”，最大并发用户数未达到 50，“文件上传下载”报错如下。

Action1.c(270):Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON" （服务器响应错误）。

```
web_custom_request("×××.com.ausso.SSO_USER_LOGON",

"URL=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
      "Method=POST",
      "Resource=0",
      "RecContentType=text/html",
"Referer=http://ecims.××××.com.cn/wps/myPortal/.cmd/cs/.ce/155/.s/322/.r/1",
      "Snapshot=t6.inf",
      "Mode=HTTP",
"Body=userid=CSTC01&userpassword=CSTCSEI&servleturl=%2Fmain.jsp&reloginservleturl=%2Flogin.jsp&username=-&systemname=%E4%B8%AD%E7%9F%B3%E5%8C%96%E5%B7%A5%E7%A8%8B%E5%BB%BA%E8%AE%BE%E4%BF%A1%E6%81%AF%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F&sub_system_code=&sub_system_name=&sdet_lang=zh",
      LAST);
```

- 真实业务测试环境：案例“项目管理”，最大并发用户数未达到 50，报错如下。

Action1.c(269):Error -26627: HTTP Status-Code=404 (Not Found) for "http://ecims.××××.com.cn/myPortal/.cmd/cs/.ce/155/.s/322/.r/1" （请求的资源未找到）。

```
web_submit_data("j_security_check",
      "Action=http://ecims.××××.com.cn/j_security_check",
      "Method=POST",
      "RecContentType=text/html",
      "Referer=http://ecims.××××.com.cn/ssoBody.jsp",
      "Mode=HTTP",
      ITEMDATA,
```

```
        "Name=j_username", "Value=CSTC01", ENDITEM,
        "Name=j_password", "Value=CSTCSEI", ENDITEM,
        LAST);
```

Action1.c(74):Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.sinopec.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON"（服务器响应错误）。

```
    web_submit_data("×××.com.ausso.SSO_USER_LOGON",
    "Action=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
        "Method=POST",
        "RecContentType=text/html",

    "Referer=http://ecims.××××.com.cn/wps/myPortal/.cmd/cs/.ce/155/.s/322/.r/1",
        "Snapshot=t24.inf",
        "Mode=HTTP",
        ITEMDATA,
        "Name=userid", "Value=CSTC01", ENDITEM,
        "Name=userpassword", "Value=CSTCSEI", ENDITEM,
        "Name=servleturl", "Value=/main.jsp", ENDITEM,
        "Name=reloginservleturl", "Value=/login.jsp", ENDITEM,
        "Name=username", "Value=-", ENDITEM,
        "Name=systemname", "Value=***********?", ENDITEM,
        "Name=sub_system_code", "Value=", ENDITEM,
        "Name=sub_system_name", "Value=", ENDITEM,
        "Name=sdet_lang", "Value=zh", ENDITEM,
        LAST);
```

Action1.c(196): Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process"（服务器响应错误）。

```
    web_custom_request("ECIMS.DT.BusinessProcess.COM_NEED_DO_process",
    "URL=http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process",
    "Method=POST",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
    "Snapshot=t9.inf",
    "Mode=HTTP",
    "Body=date=&where=+and+need_type+in+%28%27%CF%B5%CD%B3%D0%B4%C8%EB%27%2C
%27%B9%A4%D7%F7%C8%D5%B3%CC%27%29+and+status+%3D+%27%B4%FD%B4%A6%C0%ED
%27+and+substr%28alert_begin%2C1%2C10%29%3C%3D%272004-05-14%27+and+SUBSTR%28ALERT_
end%2C1%2C10%29%3E%3D%272004-05-14%27+and+EMP_ID+%3D%27CSTC01%27+order+by+ALE_
RT_BEGIN&act_type=select&userid=CSTC01&servleturl=%2Fhr%2Fcom_need_do_db_redirect_main.jsp&
```

```
StartRow=0&RowPerPage=30000",
    LAST);
```

- 真实业务测试环境：案例“工作记事”，最大并发用户数未达到 50，报错如下。

Action1.c(43):Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.××-××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON"（服务器响应错误）。

```
web_submit_data("×××.com.ausso.SSO_USER_LOGON",
"Action=http://ecims.××××.com.cn/servlet/×××.com.ausso.SSO_USER_LOGON",
    "Method=POST",
    "RecContentType=text/html",

"Referer=http://ecims.××××.com.cn/wps/myPortal/.cmd/cs/.ce/155/.s/322/.r/1",
    "Snapshot=t6.inf",
    "Mode=HTTP",
    ITEMDATA,
    "Name=userid", "Value=CSTC01", ENDITEM,
    "Name=userpassword", "Value=CSTCSEI", ENDITEM,
    "Name=servleturl", "Value=/main.jsp", ENDITEM,
    "Name=reloginservleturl", "Value=/login.jsp", ENDITEM,
    "Name=username", "Value=-", ENDITEM,
    "Name=systemname", "Value=**************?", ENDITEM,
    "Name=sub_system_code", "Value=", ENDITEM,
    "Name=sub_system_name", "Value=", ENDITEM,
    "Name=sdet_lang", "Value=zh", ENDITEM,
    LAST);
```

Action1.c(242):Error -26627: HTTP Status-Code=404 (Not Found) for "http://ecims.××××.com.cn/myPortal/.cmd/cs/.ce/155/.s/322/.r/1"（请求的资源未找到）。

```
web_submit_data("j_security_check",
    "Action=http://ecims.××××.com.cn/j_security_check",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=http://ecims.××××.com.cn/ssoBody.jsp",
    "Mode=HTTP",
    ITEMDATA,
    "Name=j_username", "Value=CSTC01", ENDITEM,
    "Name=j_password", "Value=CSTCSEI", ENDITEM,
    LAST);
```

Action1.c(393):Error -26612: HTTP Status-Code=500 (Internal Server Error) for "http://ecims.×-

×××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process?userid=CSTC01&user-name=CSTC01&systemname=%C8%D5%B3%A3%B0%EC%B9%AB&sub_system_code=hr&sub_system_name=%B9%A4%D7%F7%BC%C7%CA%C2&sdet_lang=zh&data_range=OWNER&date=2004-05-09&where=+and+need_type+%3D+%27%B9%A4%D7%F7%BC%C7%CA%C2%27+and+substr%28alert_begin%2C1%2C10%29%3C%3D%272004-05-09%27+and+SUBSTR%28ALERT_end%2C1%2C10%29%3E%3D%272004-05-09%27+order+by+ALERT_BEGIN&act_type=jsselect&userid=CSTC01&servleturl=%2Fhr%2Fcom_need_do_js_redirect.jsp&StartRow=0&RowPerPage=30000"（服务器响应错误）。

```
web_url("ECIMS.DT.BusinessProcess.COM_NEED_DO_process_2",
"URL=http://ecims.××××.com.cn/servlet/ECIMS.DT.BusinessProcess.COM_NEED_DO_process?userid=CSTC01&username=CSTC01&systemname=%C8%D5%B3%A3%B0%EC%B9%AB&sub_system_code=hr&sub_system_name=%B9%A4%D7%F7%BC%C7%CA%C2&sdet_lang=zh&data_range=OWNER&date=2004-05-09&where=+and+need_type+%3D+%27%B9%A4%D7%F7%BC%C7%CA%C2%27+and+substr%28alert_begin%2C1%2C10%29%3C%3D%272004-05-09%27+and+SUBSTR%28ALERT_end%2C1%2C10%29%3E%3D%272004-05-09%27+order+by+ALERT_BEGIN&act_type=jsselect&userid=CSTC01&servleturl=%2Fhr%2Fcom_need_do_js_redirect.jsp&StartRow=0&RowPerPage=30000",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://ecims.××××.com.cn/hr/com_need_do_js_choose.jsp",
        "Snapshot=t14.inf",
        "Mode=HTTP",
        LAST);
```

（2）响应时间

- 对比测试环境 1 和对比测试环境 2。

① 案例“制度文档”，检查点“信息上传”占总响应时间（包括检查点“查询所有”、“信息上传”、“文件下载”）的60%，约为60s。

② 案例“项目管理”，检查点“新增项目”占总响应时间（包括检查点“项目选择”、“新增项目”、“条件查询”）的65%，约为68s。

③ 案例“工作记事”，检查点“新增记事”占总响应时间（包括检查点“条件查询”、“新增记事”）的70%，约为35s。

- 业务操作环境。

下面以业务操作机房环境“项目管理”为例，进行页面组件明细分析。

① 页面组件响应时间，如图20-10和图20-11所示。



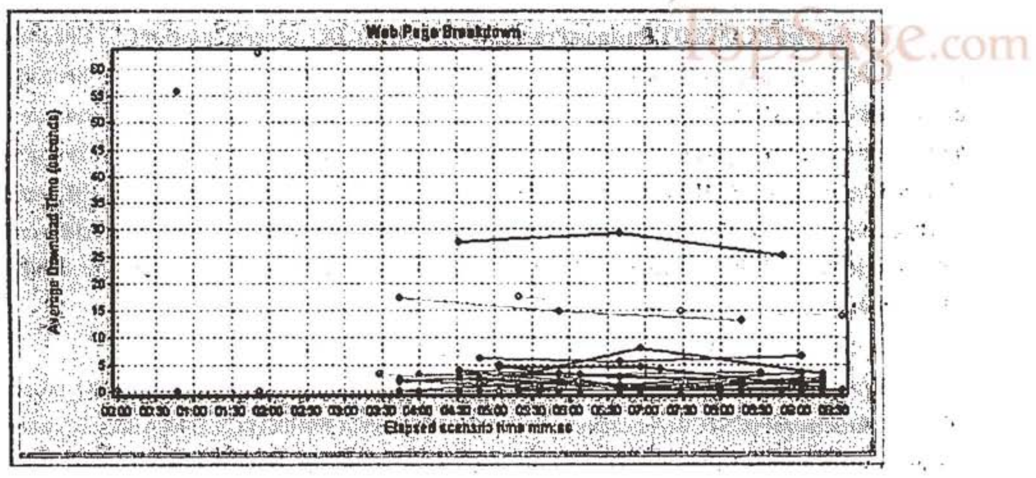


图 20-10 页面组件响应时间 I

Color	Scale	Measurement	Min.	Ave.	Max.	SD
	1	ecims.s.../Portal/.cmd/li (main URL):vuser_init_Transaction -> ecims.s.../Portal/.cmd/li (main URL)	62.851	62.851	62.851	0.001
	1	ecims....om.cn/wps/Portal (main URL):vuser_init_Transaction -> ecims....om.cn/wps/Portal (main URL)	55.867	55.867	55.867	0
	1	ecims.si...E_INFO_process (main URL):Action1_Transaction -> 项目选择 -> ecims.si...E_INFO_process (main URL)	25.14	27.35	29.248	1.692
	1	ecims.si...E_INFO_process (main URL):Action1_Transaction -> 新增项目 -> ecims.si...E_INFO_process (main URL)	13.972	15.478	17.531	1.504
	1	ecims.si...SSO_USER_LOGON(mainURL):Action1_Transaction -> 进入业务系统 -> ecims.si...SSO_USER_LOGON (main URL)	13.108	15.084	17.387	1.762
	1	ecims.si...P_INFO_process (main URL):Action1_Transaction -> 新增项目 -> ecims.si...P_INFO_process (main URL)	5.541	6.106	6.554	0.422

图 20-11 页面组件响应时间 II

	1	ecims.si..._doc_input.jsp (main URL):Action1_Transaction -> 新增项目 -> ecims.si..._doc_input.jsp (main URL)	1.501	4.272	8.086	2.787
	1	ecims.si...t_redirect.jsp (main URL):Action1_Transaction -> 项目选择 -> ecims.si...t_redirect.jsp (main URL)	3.115	3.255	3.475	0.158
	1	ecims.sino...ASE_INFO_process (main URL):Action1_Transaction -> 新增项目 -> ecims.sino...ASE_INFO_process (main URL)	0.203	3.131	5.658	1.947
	1	web_concurrent_end_Action1_2380:Action1_Transaction -> 新增项目 -> web_concurrent_end_Action1_2380	1.036	2.784	3.746	1.238
	1	web_concurrent_end_Action1_982:Action1_Transaction -> 项目选择 -> web_concurrent_end_Action1_982	1.598	2.761	3.984	0.975
	1	web_concurrent_end_Action1_4384:Action1_Transaction -> 新增项目 -> web_concurrent_end_Action1_4384	0.012	2.476	5.011	2.042
	1	web_concurrent_end_Action1_128:Action1_Transaction -> 进入业务系统 -> web_concurrent_end_Action1_128	1.829	2.374	3.198	0.593
	1	ecims.si...info_input.jsp (main URL):Action1_Transaction -> 新增项目 -> ecims.si...info_input.jsp (main URL)	0.583	1.743	2.376	0.821
	1	ecims.××××.../322/.r/1 (main URL):Action1_Transaction -> 进入业务系统 -> ecims.××××.../322/.r/1 (main URL)	0.697	1.622	3.273	1.17
	1	ecims.si...le_channel.jsp (main URL):Action1_Transaction -> 新增项目 -> ecims.si...le_channel.jsp (main URL)	0.491	1.51	2.469	0.808
	1	ecims.si...BED_DO_process (main URL):Action1_Transaction -> 进入业务系统 -> ecims.si...BED_DO_process (main URL)	0.289	1.418	2.188	0.816

图 20-11 （续）

② 页面组件响应时间明细，如图20-12和如图20-13所示。

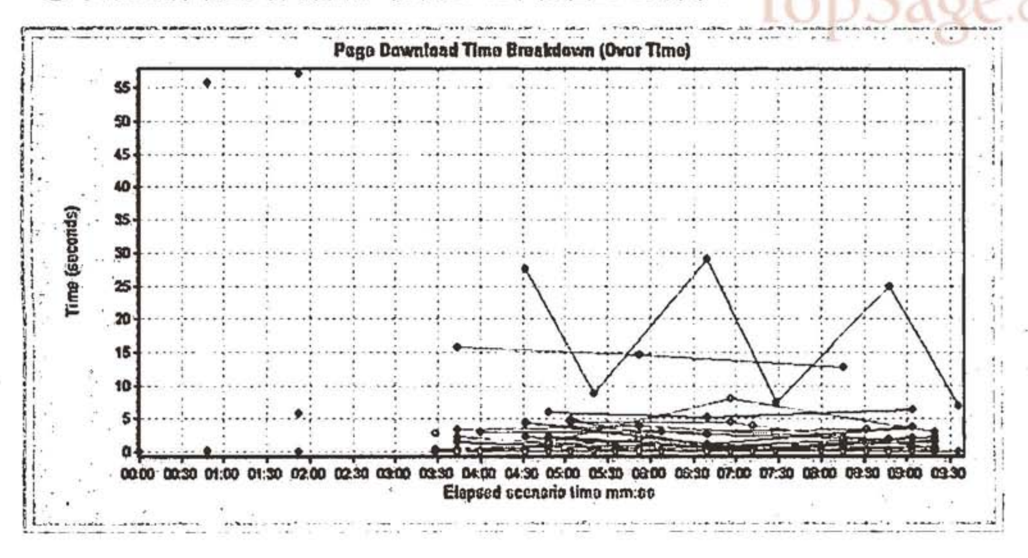


图 20-12　页面组件响应时间明细 I

Color	Scale	Measurement	Min.	Ave.	Max.	SD
	1	ecims.s.../Portal/.cmd/li (main URL).[Receive]	57.124	57.124	57.124	0
	1	ecims....om.cn/wps/Portal (main URL).[Receive]	55.729	55.729	55.729	0
	1	ecims.si...SSO_USER_LOGON (main URL).[First Buffer]	12.8	14.417	15.825	1.244
	1	ecims.si...E_INFO_process (main URL).[First Buffer]	0.124	14.227	29.119	11.129
	1	ecims.si...P_INFO_process (main URL).[First Buffer]	5.158	5.803	6.319	0.483
	1	ecims.s.../Portal/.cmd/li (main URL).[First Buffer]	5.726	5.726	5.726	0

图 20-13　页面组件响应时间明细 II

	1	ecims.si..._doc_input.jsp (main URL).[First Buffer]	1.487	4.181	7.975	2.761
	1	web_concurrent_end_Action1_128.[First Buffer]	2.123	3.126	3.957	0.758
	1	web_concurrent_end_Action1_2380.[First Buffer]	1.046	3.119	4.584	1.507
	1	ecims.sino...ASE_INFO_process (main URL).[First Buffer]	0.203	3.08	5.54	1.905
	1	ecims.si...t_redirect.jsp (main URL).[First Buffer]	2.944	3.072	3.289	0.154
	1	web_concurrent_end_Action1_982.[First Buffer]	1.8	2.891	4.31	1.051
	1	web_concurrent_end_Action1_4384.[First Buffer]	0.023	2.393	4.854	1.973
	1	ecims.si...info_input.jsp (main URL).[First Buffer]	0.58	1.61	2.178	0.729
	1	ecims.si...BED_DO_process (main URL).[First Buffer]	0.286	1.416	2.187	0.817
	1	ecims.si...le_channel.jsp (main URL).[First Buffer]	0.491	1.395	2.278	0.73
	1	ecims.××××.../322/.r/1 (main URL).[Receive]	0.003	0.96	2.873	1.353
	1	web_concurrent_end_Action1_128.[Receive]	0.161	0.81	1.929	0.794
	1	ecims.si...SSO_USER_LOGON (main URL).[Receive]	0.129	0.666	1.561	0.637

图 20-13　（续）



③ 页面组件Web服务器响应时间与网络传输延迟，如图20-14和如图20-15所示。

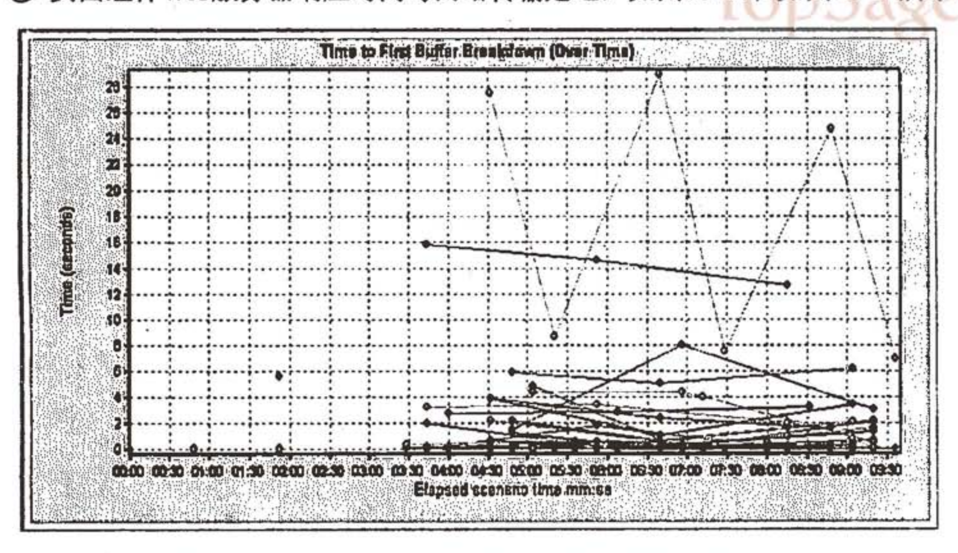


图 20-14　页面组件 Web 服务器响应时间与网络传输延迟 I

Color	Scale	Measurement	Min.	Ave.	Max.	SD
	1	ecims.si...SSO_USER_LOGON (main URL).[Server]	12.617	14.352	15.822	1.322
	1	ecims.si...E_INFO_process (main URL).[Server]	0.111	14.154	28.951	11.075
	1	ecims.si...P_INFO_process (main URL).[Server]	4.987	5.693	6.162	0.508
	1	ecims.s.../Portal/.cmd/li (main URL).[Server]	5.631	5.631	5.631	0
	1	ecims.si..._doc_input.jsp (main URL).[Server]	1.485	4.173	7.956	2.753
	1	ecims.sino...ASE_INFO_process (main URL).[Server]	0.202	3.008	5.538	1.923

图 20-15　页面组件 Web 服务器响应时间与网络传输延迟 II

	1	ecims.si...t_redirect.jsp (main URL).[Server]	2.799	2.955	3.223	0.191
	1	web_concurrent_end_Action1_128.[Server]	1.948	2.856	3.414	0.648
	1	web_concurrent_end_Action1_2380.[Server]	0.997	2.756	3.874	1.259
	1	web_concurrent_end_Action1_982.[Server]	1.634	2.626	3.845	0.917
	1	web_concurrent_end_Action1_4384.[Server]	0	2.351	4.85	1.983
	1	ecims.si...info_input.jsp (main URL).[Server]	0.579	1.608	2.177	0.729
	1	ecims.si...le_channel.jsp (main URL).[Server]	0.362	1.316	2.21	0.756
	1	ecims.si...EED_DO_process (main URL).[Server]	0.22	1.275	1.992	0.762
	1	ecims....n/inputcheck.htc (main URL).[Server]	0	0.602	1.805	0.851
	1	ecims....n/lang/zh_hr.xml (main URL).[Server]	0	0.542	1.626	0.767
	1	ecims.××××.../322/.r/1 (main URL).[Server]	0.334	0.475	0.598	0.109
	1	web_concurrent_end_Action1_3384.[Server]	0	0.403	1.208	0.569
	1	web_concurrent_end_Action1_2380.[Network]	0.049	0.363	0.71	0.271

图 20-15　（续）



④ 页面组件容量，如图20-16和如图20-17所示。

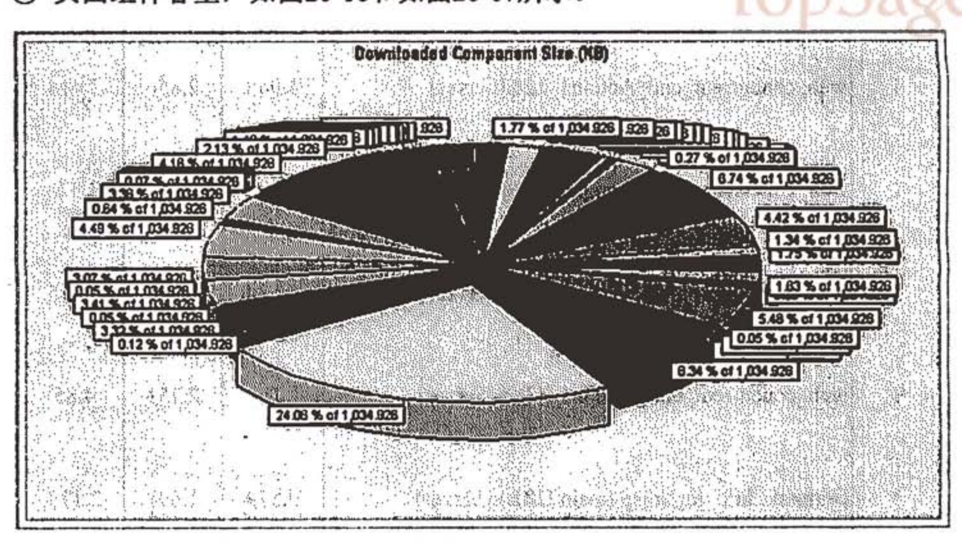


图 20-16 页面组件容量 I

Color	Scale	Measurement	Graph Min.	Graph Ave.	Graph Max.	Graph Median	Graph SD
	1	ecims.si...P_INFO_process (main URL)	248.997	248.997	248.997	248.997	0
	1	web_concurrent_end_vuser_init_165	69.724	69.724	69.724	69.724	0
	1	ecims.si...SSO_USER_LOGON (main URL)	65.602	65.602	65.602	65.602	0
	1	web_concurrent_end_Action1_128	56.692	56.692	56.692	56.692	0
	1	ecims.s.../Portal/.cmd/li (main URL)	46.432	46.432	46.432	46.432	0
	1	web_concurrent_end_Action1_982	45.778	45.778	45.778	45.778	0

图 20-17 页面组件容量 II

	1	ecims....n/lang/zh_pr.xml (main URL)	43.028	43.028	43.028	43.028	0
	1	ecims.si...E_INFO_process (main URL)	35.251	35.251	35.251	35.251	0
	1	ecims....om.cn/wps/Portal (main URL)	34.813	34.813	34.813	34.813	0
	1	ecims.si...info_input.jsp (main URL)	34.314	34.314	34.314	34.314	0
	1	ecims....fig/deeptree.htc (main URL)	32.445	32.445	32.445	32.445	0
	1	ecims.si..._doc_input.jsp (main URL)	31.74	31.74	31.74	31.74	0
	1	ecims..../login_bg002.jpg (main URL)	24.47	24.47	24.47	24.47	0
	1	ecims....n/lang/zh_hr.xml (main URL)	22.084	22.084	22.084	22.084	0
	1	ecims....es/xmglxt004.jpg (main URL)	20.053	20.053	20.053	20.053	0
	1	ecims....es/xmglxt005.jpg (main URL)	18.362	18.362	18.362	18.362	0
	1	web_concurrent_end_Action1_4384	18.1	18.1	18.1	18.1	0
	1	web_concurrent_end_Action1_3384	16.881	16.881	16.881	16.881	0
	1	ecims.××××.../322/.r/1 (main URL)	16.864	16.864	16.864	16.864	0

图 20-17　（续）



1	ecims....n/inputcheck.htc (main URL)	15.348	15.348	15.348	15.348	0
1	web_concurrent_end_Action1_45	13.883	13.883	13.883	13.883	0
1	ecims..../××××_css.css (main URL)	9.793	9.793	9.793	9.793	0
1	ecims..../sino1/title.gif (main URL)	9.021	9.021	9.021	9.021	0
1	ecims....images/title.gif (main URL)	9.021	9.021	9.021	9.021	0
1	ecims....es/title_win.gif (main URL)	8.443	8.443	8.443	8.443	0
1	ecims....fig/deeptree.xsl (main URL)	7.282	7.282	7.282	7.282	0
1	web_concurrent_end_Action1_2380	7.076	7.076	7.076	7.076	0
1	ecims....ortal/.scr/Login (main URL)	6.646	6.646	6.646	6.646	0
1	ecims..../sino1/logo.gif (main URL)	6.564	6.564	6.564	6.564	0

图 20-17 （续）

注：其他案例页面组件分析见原始测试数据（利用Loadrunner测试分析器查看，本书略）。

（3）资源占用

- Portal 服务器。

① CPU：对比测试环境1和对比测试环境2，平均值接近90%，最大值达到100%，最大值持续时间约占整个测试时间的30%；业务测试环境，CPU占用率频繁达到100%。

② 内存：使用正常，虚拟内存使用量较少。

③ 硬盘：使用正常。

- CM 服务器。

① CPU：对比测试环境1和对比测试环境2，较空闲，平均值未达到5%，最大值接近10%；业务测试环境，“登录”操作期间CPU占用率持续达到100%。

② 内存：使用正常，虚拟内存使用量较少。

③ 硬盘：使用正常。

- DB2 资源。

Portal DB2 7.2有失败SQL调用；锁数目最大值达到150，等待处理的代理数目的最大值达到1.8，没有锁超时现象；pool指标值全部为零，动态sql请求最大值为35237，静态sql请求最大值为1828。

CM DB2 8.1各项监控指标正常。

- Websphere 资源。

① 非法Session最大值达到约850（包括以数据库模式调用的session）；

② 等待锁的请求数最大值达到301590，锁等待的最大平均时间为10s；

③ 同步请求数在机房环境下最大值达到约为135；同步请求数在局域网环境下最大值达到约为35，此值建议提高为50；请求的响应时间建议低于3s；

④ JVM内存使用情况合理，约占150MB，属合理使用范围（占用率达到50%）。

2．广域网结果分析

（1）交易响应时间对比

① 数据。

27表示机房局域网，28表示A广域网。表20-35所示为交易响应时间对比。

表 20-35　交易响应时间对比

Transaction Name	Max27	Max28	2M27	2M28	512K27	512K28	56K27	56K(5*1)28
Action_Transaction	192.135	408.91	151.545	213.596	184.59	205.364	745.079	135.205
Action1_Transaction	163.269	673.091	124.848	375.883	159.383	295.568	283.232	255.62
vuser_end_Transaction	0	0	0	0	0	0	0	0
vuser_init_Transaction	197.122	472.324	191.468	206.14	187.796	211.373	201.24	125.318
文件下载	36.941	74.78	28.641	66.12	50.031	61.382	83.111	63.051
新增上传文件	33.585	167.557	23.445	81.313	32.962	79.519	101.321	99.046
新增信息上传	18.226	149.827	14.85	68.303	21.323	42.814	57.005	60.454
新增工作记事	32.524	39.79	25.155	35.043	38.226	28.054	227.864	27.836
新增项目	51.312	288.036	47.363	144.31	67.699	119.384	154.308	152.515
条件查询	2.885	5.276	2.283	6.951	2.644	5.184	4.261	6.453
查询全部集团	9.785	54.23	8.067	21.612	8.098	32.527	21.518	25.735
查询所有综合管理类	8.931	65.141	6.368	30.888	9.907	28.314	23.206	30.24
进入业务系统	39.944	304.343	41.271	145.99	38.158	117.498	62.483	66.614
进入业务系统	41.024	285.463	31.359	146.348	31.712	91.762	58.04	63.979

续表

Transaction Name	Max27	Max28	2M27	2M28	512K27	512K28	56K27	56K(5*1)28
进入工作记事	34.63	57.701	33.312	15.316	42.415	32.093	219.841	18.051
进入日常办公	1.001	1	1.001	1.001	1.002	1.001	1.001	1.07
进入系统管理	13.194	76.386	9.935	43.588	14.687	39.13	26.27	31.068
选择日期	35.081	16.633	24.451	10.19	34.778	19.599	212.601	12.691
项目查询	11.1	26.369	4.946	13.665	8.546	16.148	18.654	17.152
项目选择	25.608	75.232	23.931	38.138	30.097	42.924	35.786	36.202

② 结论。

对比局域网和广域网交易平均响应时间，同一交易在相同带宽下，广域网测试环境下的响应时间，都比局域网测试环境下的响应时间长，响应时间差见表20-36。

并发执行报错主要包括：网络传输错误、请求超时错误以及服务器响应错误。

表 20-36　交易响应时间差对比

案例 / 响应时间差	登录(init_Transaction)	交易1(Action1_Transaction)	交易2(Action1_Transaction)
不限带宽	275.202s	216.775s	509.822s
2Mb带宽	14.672s	62.051s	251.035s
512Kb带宽	23.577s	20.774s	136.185s

（2）应用网络故障定位

- 应用网络处理时间分析（Performance Overview 和 CNS Breakdown）。

① 数据。

Node（Portal服务器）：Processing 94.745 sec，占16.2%；Sending 486.735 sec，占83.22%。如图20-18所示为性能总览；如图20-19所示为CNS分解图。

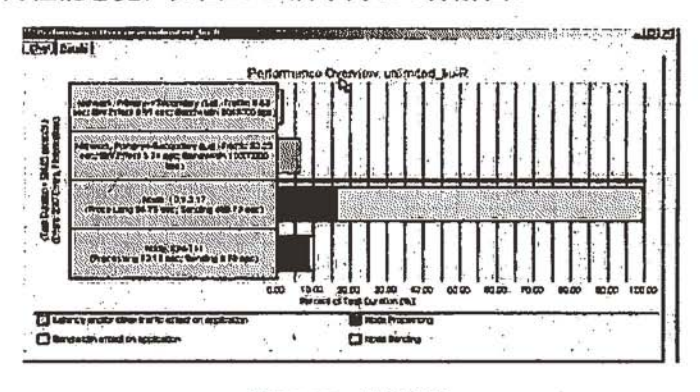


图 20-18　性能总览

② 结论。从如图 20-18 和如图 20-19 所示可知，广域网环境中网络传输时间所占比例较高，Portal 服务器发出的数据，在广域网络上的传输时间，相对服务器本身处理时间要长。网络带宽的影响和其他应用程序争用带宽的影响不是主导因素。因而，重点考察 Portal 服务器的网络发送时间。

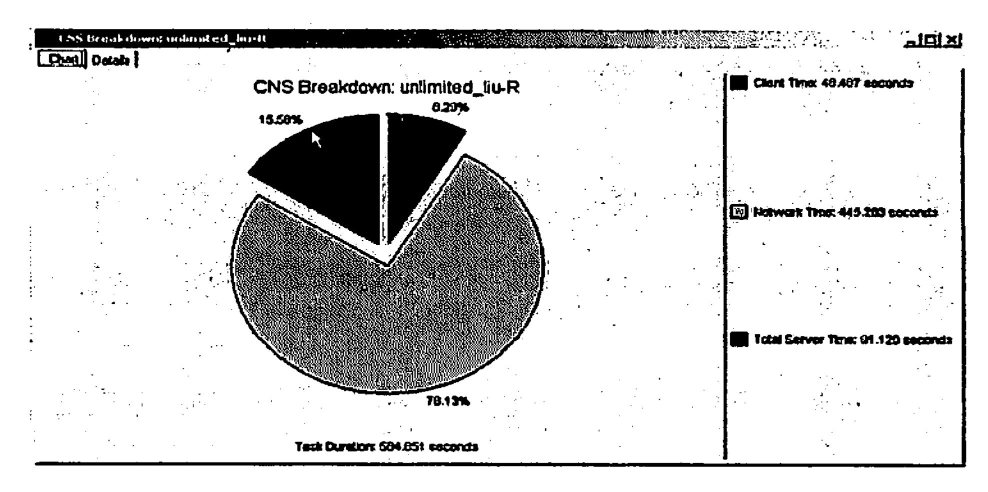


图 20-19　CNS 分解图

- 节点传输分析（Node Sending Details）。

① 数据。如图20-20所示。

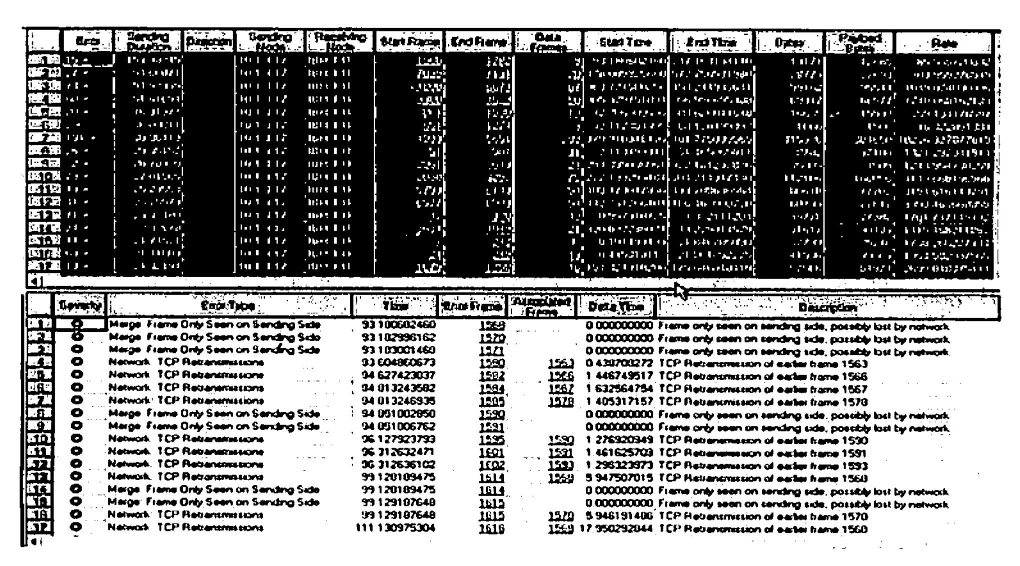


图 20-20　节点传输分析



② 结论。从如图20-20所示，Portal服务器发出的数据帧的明细，可见应用数据在广域网上存在很多帧丢失和TCP重传的现象。（Error： Frame only seen on sending side. prossibly lost by network；TCP Retransmission of earlier frame xxxx）。通过分析数据中的TCP ACK帧，可知TCP重传不是由客户端接收缓存满，而不能及时接收数据造成的。要综合考虑网络品质问题和数据发送端的有关TCP重传参数设置。

3. 包跟踪分析（PACKET TRACE）

① 数据。如图20-21所示。

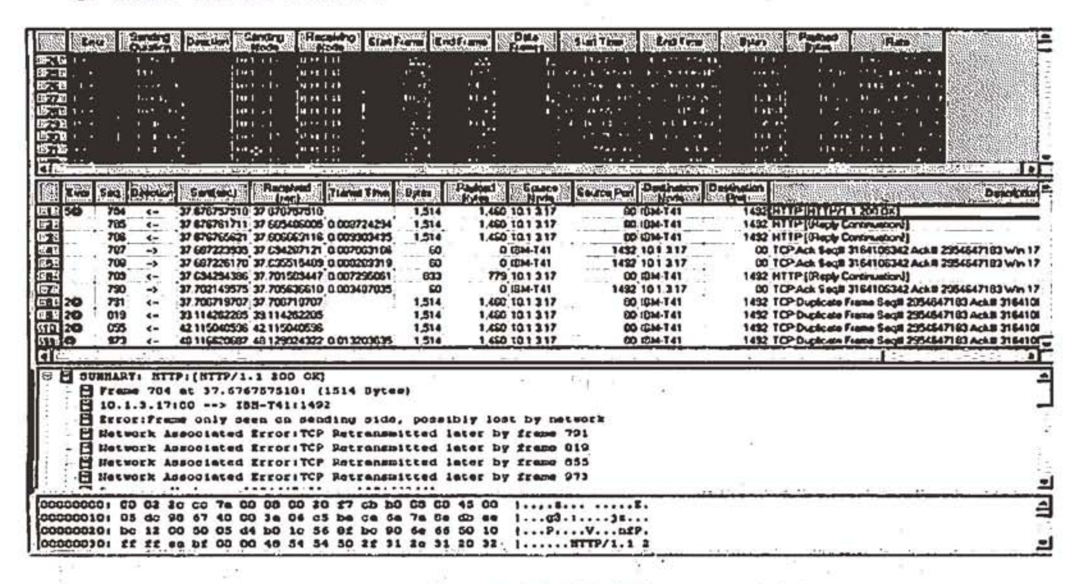


图 20-21　包跟踪分析

② 结论。如图20-21所示，针对起始序号784、结尾序号973的数据帧进行包跟踪分析，序号784的数据丢失，导致多次重传，序号791、819、855、973均为该数据帧的重传帧，且重传的延时不断递增（重传开始时间间隔为1.407542578sec、3.000778311sec、6.001580091sec）。因此造成应用数据传输时间增加。

分析出错重传的数据帧，发现字节少的帧（如60字节的TCP ACK）较少出现丢失重传，字节多的帧（如1514字节的HTTP[HTTP/1.1 200 OK]）丢失重传较多。

数据传输的包大则增加了帧的传输次数，同时增加了出错概率。

注意，本结果是A工作区的测试机处的探针，与总部的路由器后的探针数据的整合结果。

4. 应用网络性能分析

- 应用程序会话分析。

① 数据。如表20-37、如图20-22所示。

表 20-37　应用程序会话往返行程

案例	项目管理	工作记事	制度文档文件上传下载	制度文档信息上传
Portal服务器到CM服务器	290	290	290	293
负载生成器到Portal服务器	138	98	100	118

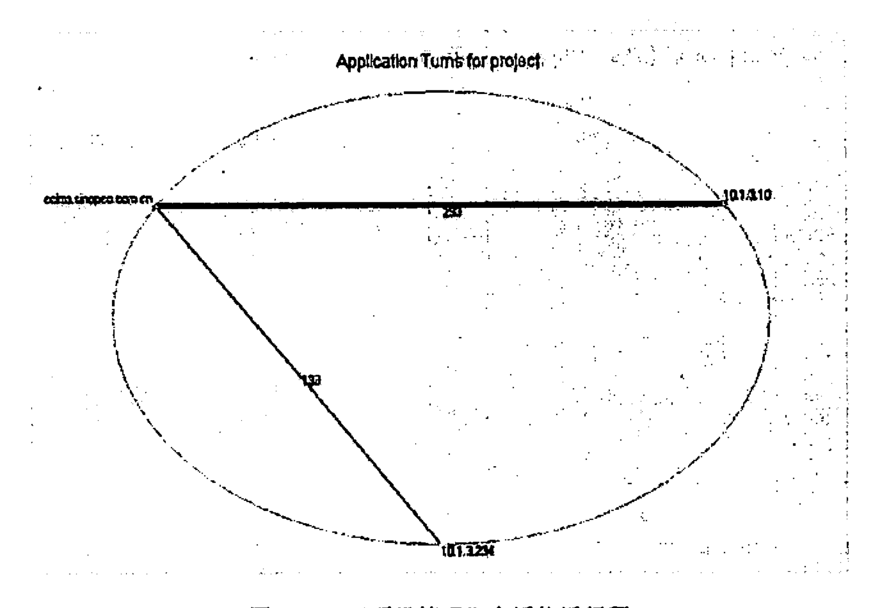


图 20-22　“项目管理”会话往返行程

② 结论。

■　表 20-37 是典型案例应用程序会话往返行程统计，如图 20-22 所示是案例“项目管理”会话往返行程测试数据。

■　针对不同的案例，Portal 服务器到 CM 服务器会话往返行程基本保持一致，可以看到系统应用会话模式基本一致，有利于系统维护。

■　负载生成器到 Portal 服务器运行在广域网环境下，往返行程次数越多，受网络品质、网络延迟影响越大。例如，“项目管理”案例达到 138 次。

③ 建议。建议降低负载生成器到Portal服务器的往返行程次数。

■　应用程序线程分析。

① 数据。

以案例“项目管理”为例进行线程分析。



- 如图 20-23 所示是运行案例“项目管理”线程性能数据。
- 如图 20-24 所示是线程之间 gap 达到 3 秒（48.4～45.6）的线程。
- 如图 20-25 所示是线程与会话（bounce）之间的关系。
- 如图 20-26 所示是线程与包（packet）之间的关系。
- 如表 20-38 所示是往返行程最大的三个线程。
- 如表 20-39 所示是往返行程最大的三个线程的主要性能数据。
- 如表 20-40 所示是线程之间 gap 达到 3 秒。
- 如表 20-41 所示是线程的请求与回应。

图 20-23 “项目管理”线程性能数据

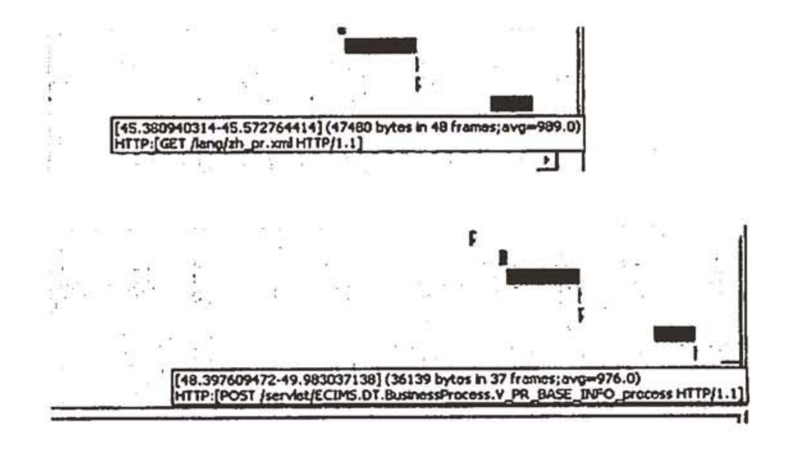


图 20-24 线程之间 gap 达到 3 秒（48.4～45.6）的线程

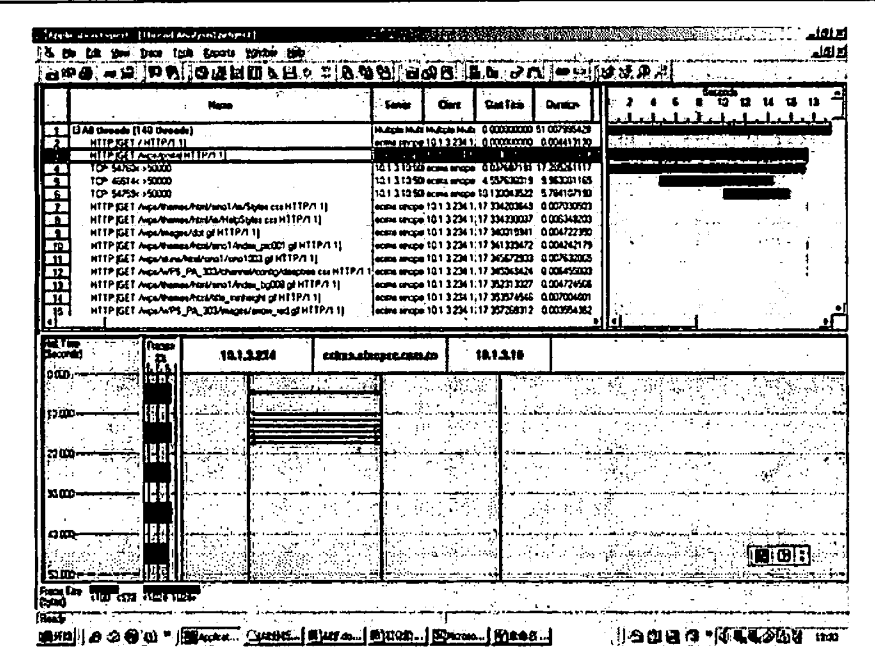


图 20-25　线程与会话（bounce）之间的关系

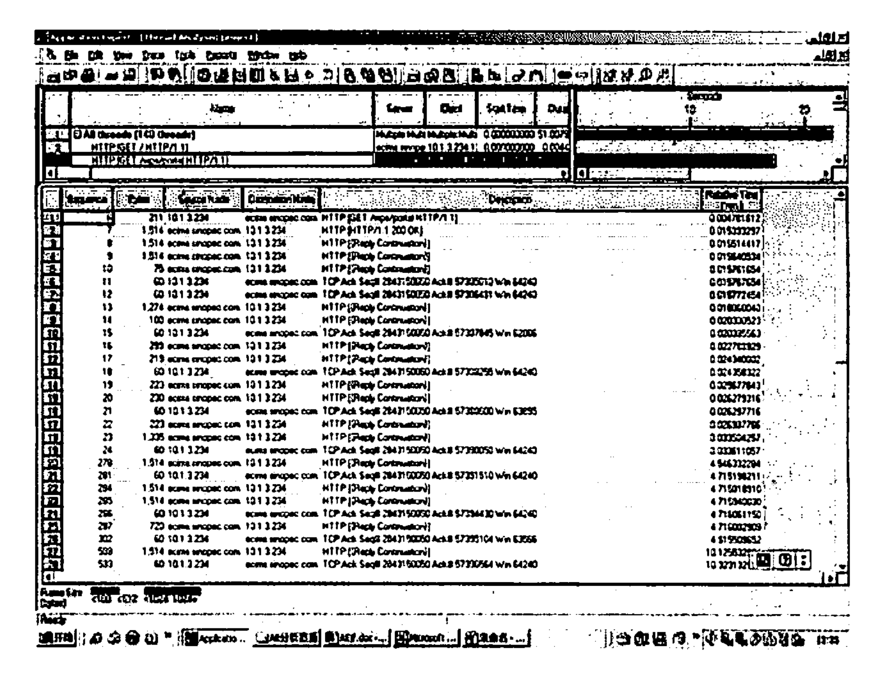


图 20-26　线程与包（packet）之间的关系



表 20-38 往返行程最大的三个线程

Name	Protocol	App. Turns	TCP Turns	Bytes/App. Turn	Frames/App. Turn
HTTP:[GET / HTTP/1.1]	HTTP	126	0	493	2.5
HTTP:[GET /wps/Portal HTTP/1.1]	HTTP	120	22	39291	62
TCP: 54760<->50000	TCP	44	4	1783	3.1

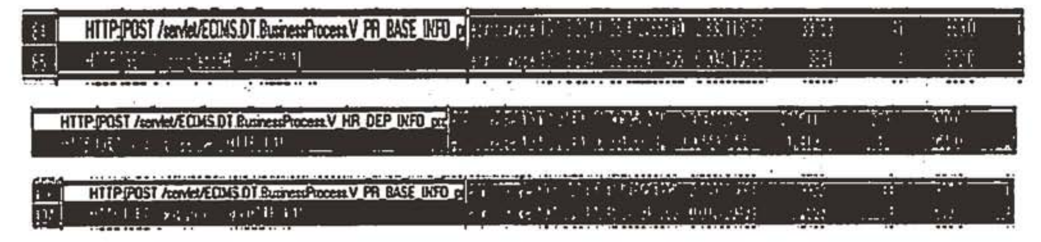
表 20-39 往返行程最大的三个线程主要性能数据

	Name	Server	Client	Start Time	Duration	Bytes	Frames	Avg. Frame Size
1	⊟All threads (140 threads)	Multiple:Multiple	Multiple:Multiple	0.000000000	51.007995428	1982595	2581	768.1
2	HTTP:[GET / HTTP/1.1]	ecims.sinopec.com.cn:80	10.1.3.234:1242	0.000000000	0.004413130	986	5	197.0
3	HTTP:[GET /wps/portal HTTP/1.1]	ecims.sinopec.com.cn:80	10.1.3.234:1242	0.004781612	17.328785464	39291	62	633.0
4	TCP: 54760<->50000	10.1.3.10:50000	ecims.sinopec.com.cn:54760	0.037687191	17.285261117	224660	390	576.0

表 20-40 线程之间 gap 达到 3 秒

	Name	Server / Start Time / Duration	Bytes
97	HTTP:[GET /lang/zh_cn.xml HTTP/1.1]	[illegible]	[illegible]
98	[illegible]	[illegible]	[illegible]
138	HTTP:[GET /lang/zh_cn.xml HTTP/1.1]	[illegible]	[illegible]
139	[illegible]	[illegible]	[illegible]

表 20-41 线程的请求与回应



	Name	Data
	HTTP:[POST /servlet/ECIMS.DT.BusinessProcess.V_PR_BASE_INFO_p	[illegible]
	[illegible]	[illegible]
	HTTP:[POST /servlet/ECIMS.DT.BusinessProcess.V_HR_DEP_INFO_p	[illegible]
	[illegible]	[illegible]
	HTTP:[POST /servlet/ECIMS.DT.BusinessProcess.V_PR_BASE_INFO_p	[illegible]
	[illegible]	[illegible]

② 结论。

- 表 20-38 中列举了应用往返行程（App. Turns）最大的三个线程，这三个线程受网络品质、网络延迟影响最大。
- 表 20-39 中列举了往返行程最大的三个线程主要性能数据，其中线程“HTTP:[GET /wps/Portal HTTP/1.1]”与“TCP: 54760<->50000”的持续时间最长，并且线程“TCP: 54760<->50000”发出和接收的字节数和帧数最大。
- 如图 20-24 及表 20-40 中显示了线程之间系统间歇（gap）达到 3 秒，可见系统空闲时间在控制范围之内，基本可以接收。
- 表 20-41 显示了线程的请求与回应，线程的请求与回应基本在 1 秒钟之内，但

也有约 3 秒钟的响应时间，例如表 20-41 所示。

- 本案例中不存在相同的请求再次发出的现象。
- 如图 20-25 和图 20-26 线程与会话（bounce）以及包（packet）之间的关系，包之间的相对时间（relative time）决定了包传输的效率。

③ 建议。对关键线程进行相关调优，有必要参考包数据。

注意，其他案例数据见测试原始数据。

- 响应时间分析。

① 数据。以案例“项目管理”为例进行响应时间分析，其他案例柱状图趋势与此相同。

如图20-27所示是运行案例“项目管理”响应时间性能数据。

- 10.1.3.234:表示客户端与 Portal 服务器会话响应时间；
- 10.1.3.10: 表示 CM 服务器（起始点）与 Portal 服务器（终止点）会话响应时间；
- ecims.××××: 表示 Portal 服务器（起始点）与 CM 服务器（终止点）会话响应时间。

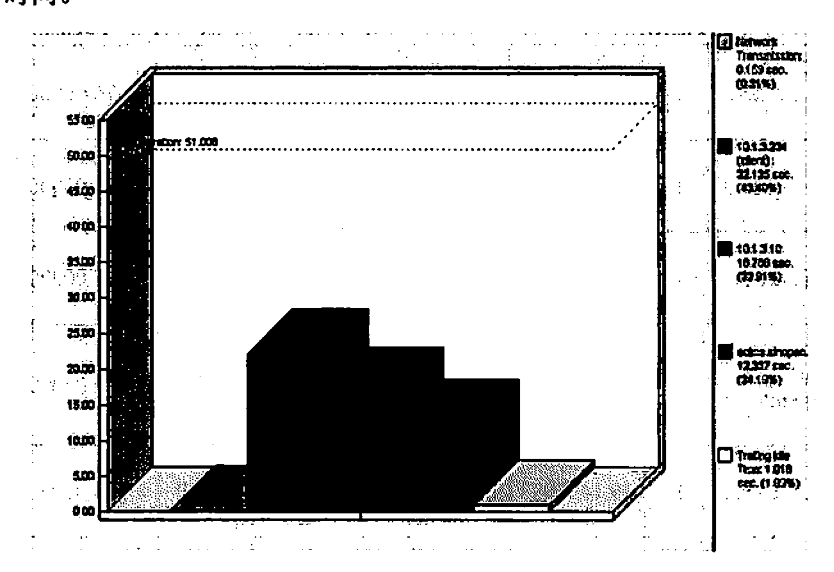


图 20-27　运行案例“项目管理”响应时间性能数据

② 结论。Portal服务器与CM服务器通信双向性清晰。

注意，其他案例数据见测试原始数据。

- 交易峰值分析。

① 数据。以案例“项目管理”为例进行交易峰值分析。数据如图20-28和表20-42所示。



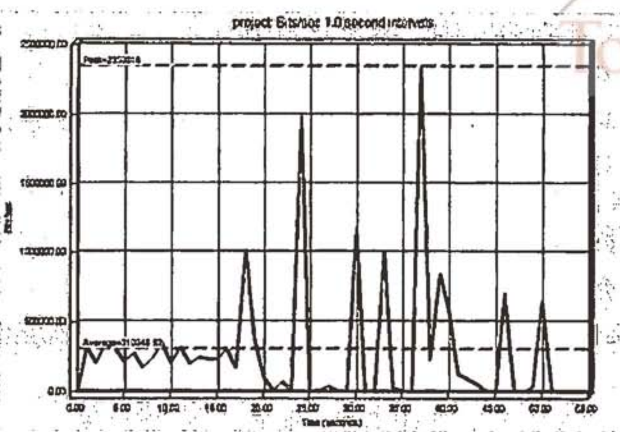


图 20-28 案例“项目管理”交易峰值性能数据

表 20-42 交易峰值对应的相关线程

Name	App. Turns	Start Time	Duration	Bytes	Frames	Protocol	Bytes/App. Turn	Frames/ App. Turn
HTTP:[GET /channel/images/inner.gif HTTP/1.1]	1	39.1651648	0.19883	2376	3	HTTP	2,376.00000	3.00000
HTTP:[GET /channel/images/go2.gif HTTP/1.1]	1	39.1615300	0.00322	1631	2	HTTP	1,631.00000	2.00000
HTTP:[GET /channel/config/channel.xml HTTP/1.1]	1	39.1577812	0.00346	3158	4	HTTP	3,158.00000	4.00000
HTTP:[GET /channel/config/ deeptreeconfig.xml HTTP/1.1]	1	39.1541629	0.00322	1527	2	HTTP	1,527.00000	2.00000
HTTP:[GET /channel/config/deeptree.xsl HTTP/1.1]	1	39.1492933	0.00440	8774	10	HTTP	8,774.00000	10.00000
HTTP:[GET /channel/config/deeptree.htc HTTP/1.1]	1	39.1411981	0.00787	35876	35	HTTP	35,876.00000	35.00000
HTTP:[GET /css.css HTTP/1.1]	1	39.1348691	0.12901	10638	12	HTTP	10,638.00000	12.00000
HTTP:[GET /channel/config/deeptree.css HTTP/1.1]	1	39.1348040	0.00620	10678	11	HTTP	10,678.00000	11.00000

续表

Name	App. Turns	Start Time	Duration	Bytes	Frames	Protocol	Bytes/App. Turn	Frames/ App. Turn
HTTP:[POST /channel/sele_channel.jsp HTTP/1.1]	1	38.1387967	0.99536	17628	25	HTTP	17,628.00000	25.00000
HTTP:[GET /lang/zh_pr.xml HTTP/1.1]	1	38.1336849	0.00486	47300	47	HTTP	47,300.00000	47.00000
HTTP:[GET /lang/zh_com.xml HTTP/1.1]	1	38.1324724	0.00087	2440	4	HTTP	2,440.00000	4.00000
HTTP:[POST /pr/pr_proj_info_doc_input.jsp HTTP/1.1]	1	38.0326877	0.09958	38271	47	HTTP	38,271.00000	47.00000
HTTP:[GET /images/list.gif HTTP/1.1]	1	37.0230001	0.13789	1564	3	HTTP	1,564.00000	3.00000
HTTP:[GET /images/open.gif HTTP/1.1]	1	37.0095400	0.00262	1471	2	HTTP	1,471.00000	2.00000
HTTP:[GET /images/fold.gif HTTP/1.1]	1	37.0057256	0.00343	1471	2	HTTP	1,471.00000	2.00000
HTTP:[GET /lang/zh_hr.xml HTTP/1.1]	1	36.9988079	0.00670	24761	25	HTTP	24,761.00000	25.00000
HTTP:[GET /lang/zh_com.xml HTTP/1.1]	1	36.9976042	0.00090	2445	4	HTTP	2,445.00000	4.00000
HTTP:[GET /js/data/hr_dep_info.js HTTP/1.1]	1	36.9919613	0.16893	7796	9	HTTP	7,796.00000	9.00000
HTTP:[GET /js/prog/treetrue.js HTTP/1.1]	1	36.9918668	0.00552	12419	13	HTTP	12,419.00000	13.00000
HTTP:[GET /images/menu005.gif HTTP/1.1]	1	23.4434742	0.19843	1430	3	HTTP	1,430.00000	3.00000
HTTP:[GET /images/title.gif HTTP/1.1]	1	23.4395601	0.00365	10555	11	HTTP	10,555.00000	11.00000



续表

Name	App. Turns	Start Time	Duration	Bytes	Frames	Protocol	Bytes/App. Turn	Frames/ App. Turn
HTTP:[GET /images/xmglxt004.jpg HTTP/1.1]	1	23.4343376	0.00474	22499	23	HTTP	22,499.00000	23.00000
HTTP:[GET /images/xmglxt001.gif HTTP/1.1]	1	23.4308327	0.00324	5520	7	HTTP	5,520.00000	7.00000
HTTP:[GET /images/xmglxt005.jpg HTTP/1.1]	1	23.4251437	0.00547	20600	20	HTTP	20,600.00000	20.00000
HTTP:[GET /images/line.gif HTTP/1.1]	1	23.4222651	0.00249	1267	2	HTTP	1,267.00000	2.00000
HTTP:[GET /images/xmglxt002.gif HTTP/1.1]	1	23.4184719	0.00337	5712	7	HTTP	5,712.00000	7.00000
HTTP:[GET /lang/zh_com.xml HTTP/1.1]	1	23.4173282	0.00085	2376	4	HTTP	2,376.00000	4.00000
HTTP:[GET /images/img002.gif HTTP/1.1]	1	23.4140401	0.00287	1334	2	HTTP	1,334.00000	2.00000
HTTP:[POST /servlet/ECIMS.DT. BusinessProcess.COM_NEED_DO_process HTTP/1.1]	1	23.1636286	0.24985	2074	5	HTTP	2,074.00000	5.00000
HTTP:[GET /images/bg003.gif HTTP/1.1]	1	23.1492866	0.00261	1320	2	HTTP	1,320.00000	2.00000
HTTP:[GET /images/bg004.gif HTTP/1.1]	1	23.1463890	0.00251	1300	2	HTTP	1,300.00000	2.00000
HTTP:[GET /inputcheck.htc HTTP/1.1]	1	23.1418750	0.00429	17305	17	HTTP	17,305.00000	17.00000

续表

Name	App. Turns	Start Time	Duration	Bytes	Frames	Protocol	Bytes/App. Turn	Frames/ App. Turn
HTTP:[GET /lang/zh_com.xml HTTP/1.1]	1	23.1382329	0.00290	2498	4	HTTP	2,498.00000	4.00000
HTTP:[GET /images/background01.gif HTTP/1.1]	1	23.1353071	0.00255	1332	2	HTTP	1,332.00000	2.00000
HTTP:[GET /××××_css.css HTTP/1.1]	1	23.1312876	0.00377	11344	11	HTTP	11,344.00000	11.00000
HTTP:[GET /images/arrow_left.gif HTTP/1.1]	1	23.1282289	0.11311	1371	3	HTTP	1,371.00000	3.00000
HTTP:[GET /js/other/bottom.js HTTP/1.1]	1	23.1274885	0.00345	3238	4	HTTP	3,238.00000	4.00000
HTTP:[GET /images/img001.gif HTTP/1.1]	1	23.1239716	0.00325	2468	5	HTTP	2,468.00000	5.00000
HTTP:[GET /main_info.jsp HTTP/1.1]	1	23.1239015	0.00411	4323	8	HTTP	4,323.00000	8.00000
HTTP:[GET /images/logo.gif HTTP/1.1]	1	23.1183582	0.00534	7870	8	HTTP	7,870.00000	8.00000
HTTP:[GET /js/prog/public.js HTTP/1.1]	1	23.1182782	0.00569	15252	16	HTTP	15,252.00000	16.00000
HTTP:[GET /js/other/mode_mk.js HTTP/1.1]	1	23.1110316	0.00732	12646	13	HTTP	12,646.00000	13.00000
HTTP:[GET /js/other/menuctrl.js HTTP/1.1]	1	23.1109705	0.00711	20990	22	HTTP	20,990.00000	22.00000
HTTP:[POST /servlet/×××.com.ausso.SSO_USER_LOGON HTTP/1.1]	1	22.0405258	1.06975	72981	84	HTTP	72,981.00000	84.00000

② 结论：

■ 交易最高峰值达到 2.4M,1M 以下的峰值分布较均匀。



■ 交易最高峰值 2.4M 所对应的执行线程是调优的关键。

③ 建议：降低交易最高峰值，调整这一时刻某些线程执行的起始时间。

- 响应时间预测分析。

结论：以案例“项目管理”为例进行响应时间预测分析。

- 如图 20-29 所示为在 56Kb 带宽上运行时，预测的响应时间性能数据。
- 如图 20-30 所示为在 512Kb 带宽上运行时，预测的响应时间性能数据。
- 如图 20-31 所示为在 2Mb 带宽上运行时，预测的响应时间性能数据。
- 如图 20-32 所示为在 10Mb 带宽上运行时，预测的响应时间性能数据。

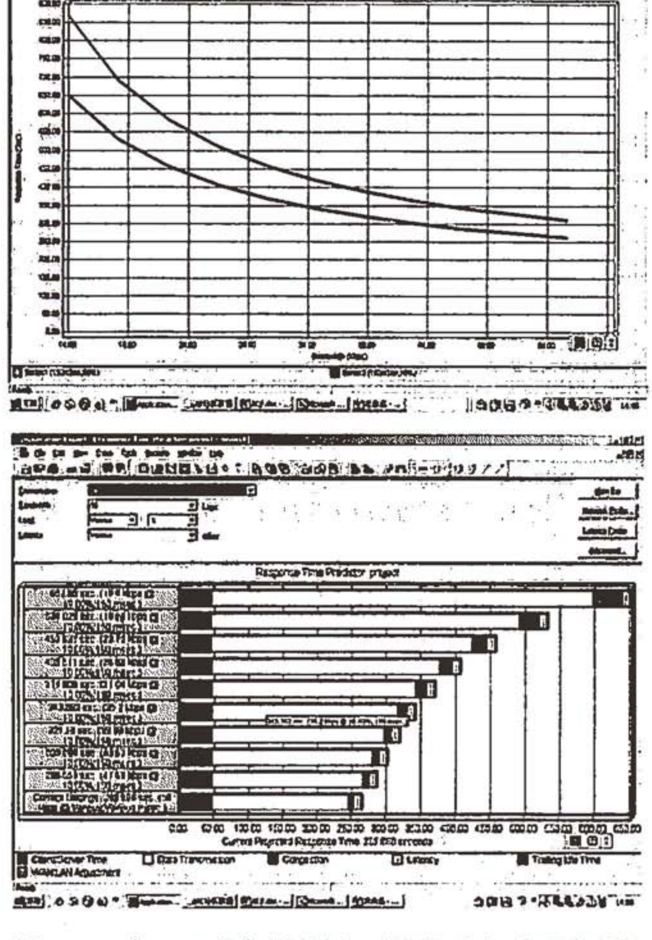


图 20-29 在 56Kb 带宽上运行时，预测的响应时间性能数据

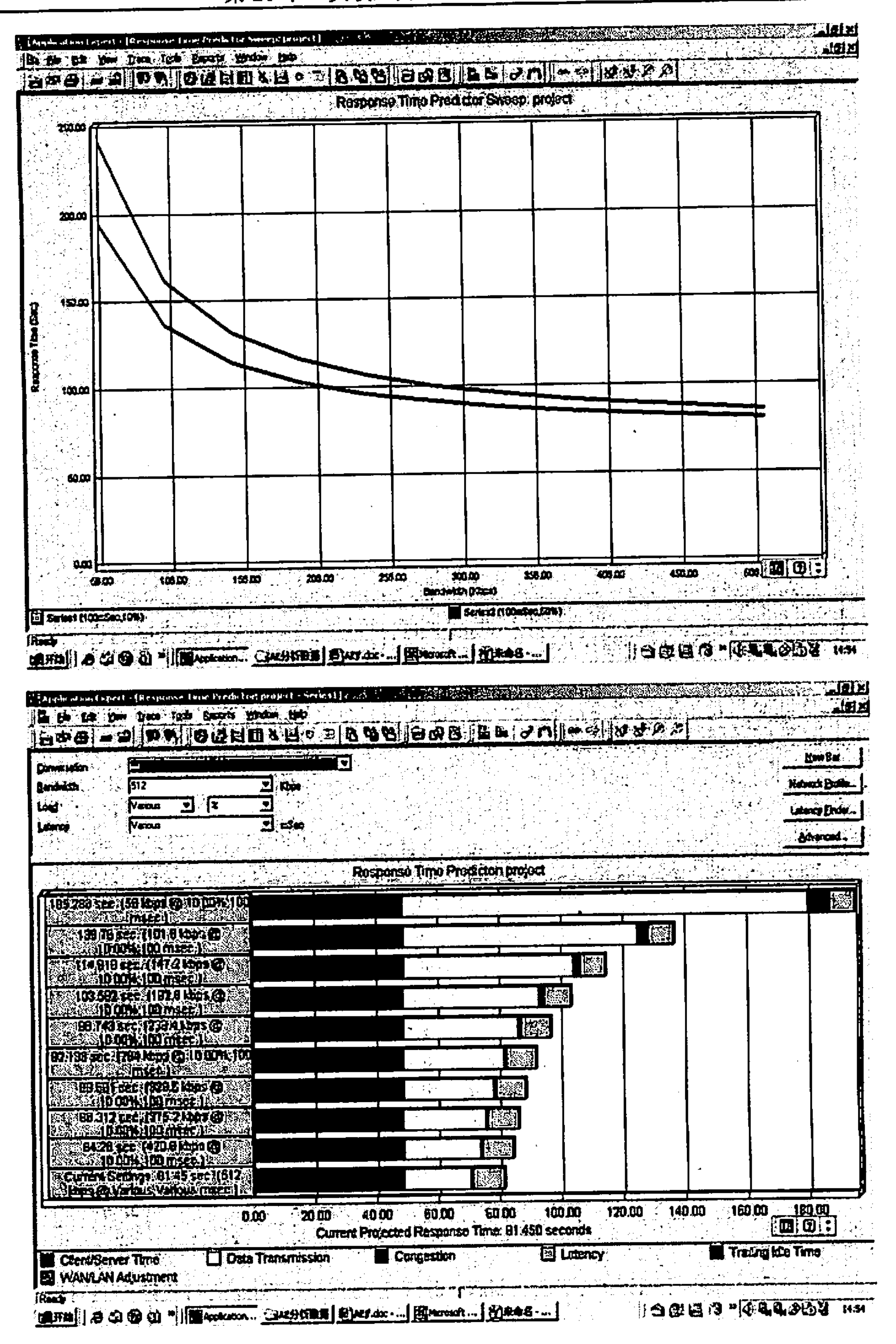


图 20-30　在 512Kb 带宽上运行时，预测的响应时间性能数据



图 20-31 在 2Mb 带宽上运行时，预测的响应时间性能数据

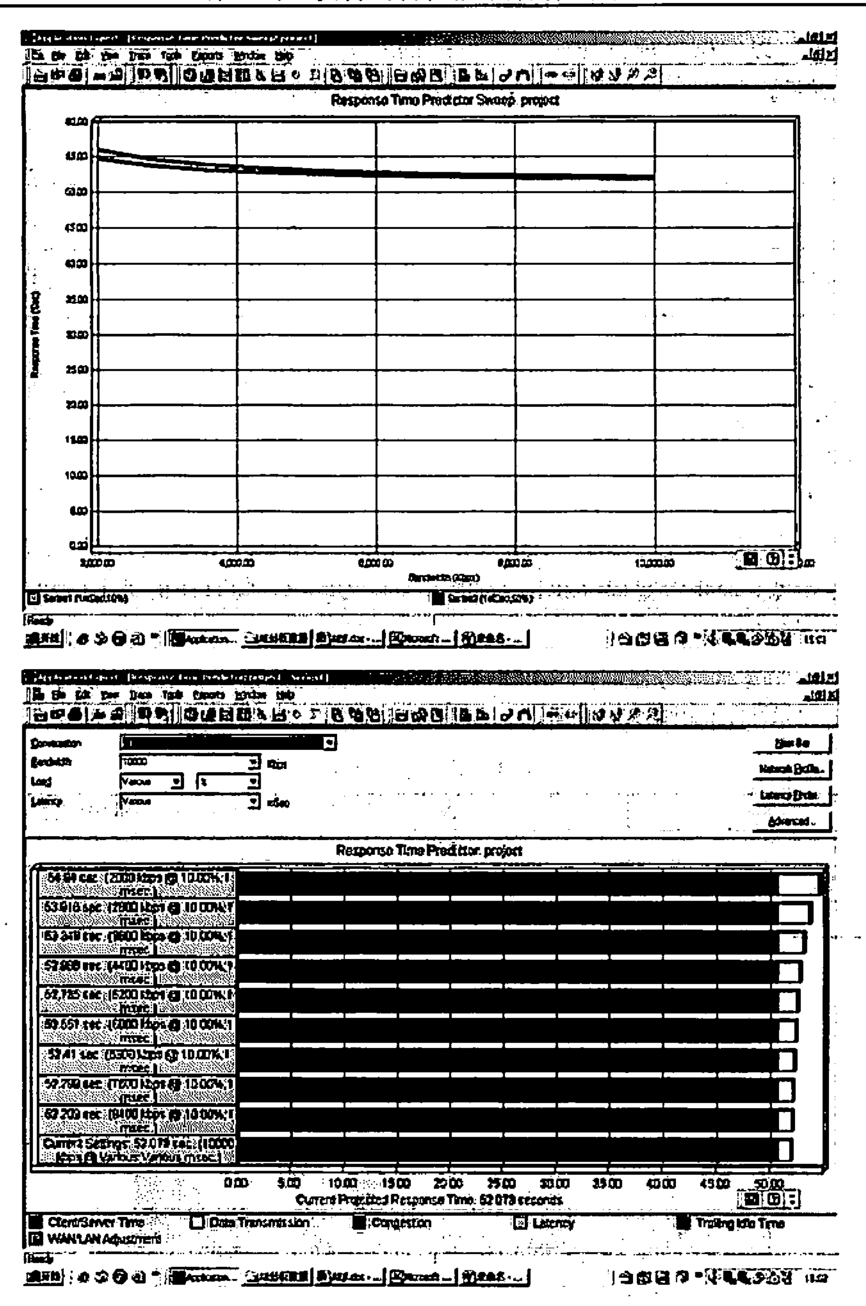


图 20-32　在 10Mb 带宽上运行时，预测的响应时间性能数据

- 帧数据分析。

① 数据。以案例“项目管理”、“工作记事”为例进行帧数据分析。帧数据统计如表20-43所示。

表 20-43 帧数据统计

帧 / 案例	小于100（字节）	大于100并且小于512（字节）	大于512并且小于1024（字节）	大于1024（字节）
项目管理	869	345	267	1097
工作记事	1567	391	178	2508

② 结论：小于100字节和大于1024字节的帧占的比例最高。

5. 网络品质监控

（1）网络流量分析。

① 数据。

下述流量分布图，分别是在14:58~15:15、15:41~16:08、16:55~17:16时刻在A工作区进行的不限带宽、2Mb带宽和512Kb带宽的并发压力测试，同时在机房出口和A工作区出口进行监测得到的。

- 不限带宽（服务器时间 14:58~15:15）。其流量分布如表 20-44 所示。A 工作区流量分布图如图 20-33 所示，机房出口流量分布图如图 20-34 所示。

表 20-44 不限带宽流量分布

	A	机房出口
应用流量（MB）	10.51	17.01
总流量（MB）	209.57	966.52
百分比	5.03%	1.76%

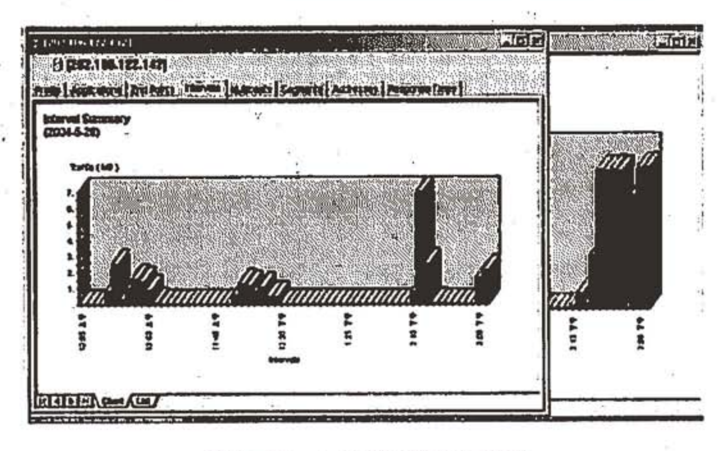


图 20-33 A 工作区流量分布图

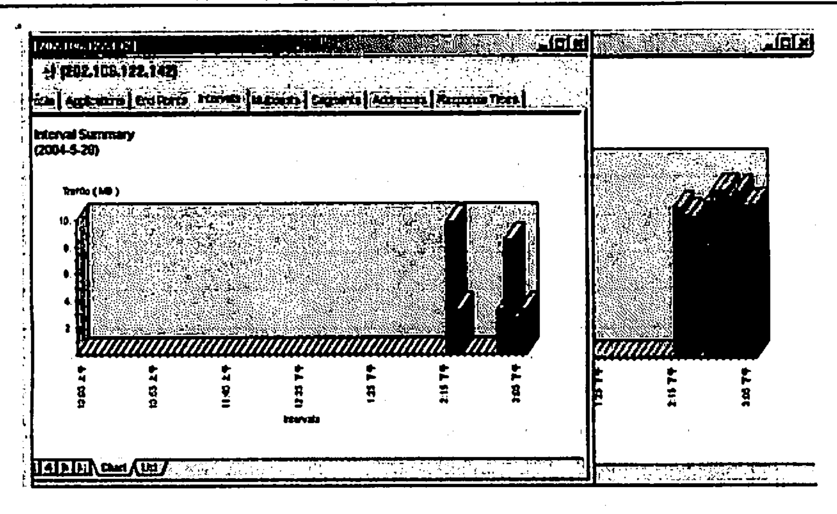


图 20-34　机房出口流量分布图

- 2Mb 带宽（服务器时间 15:41~16:08），其流量分布如表 20-45 所示，A 工作区流量分布图如图 20-35 所示。

表 20-45　2M 带宽流量分布

	A	机房出口
应用流量（MB）	32.84	
总流量（MB）	178.43	
百分比	18.4%	

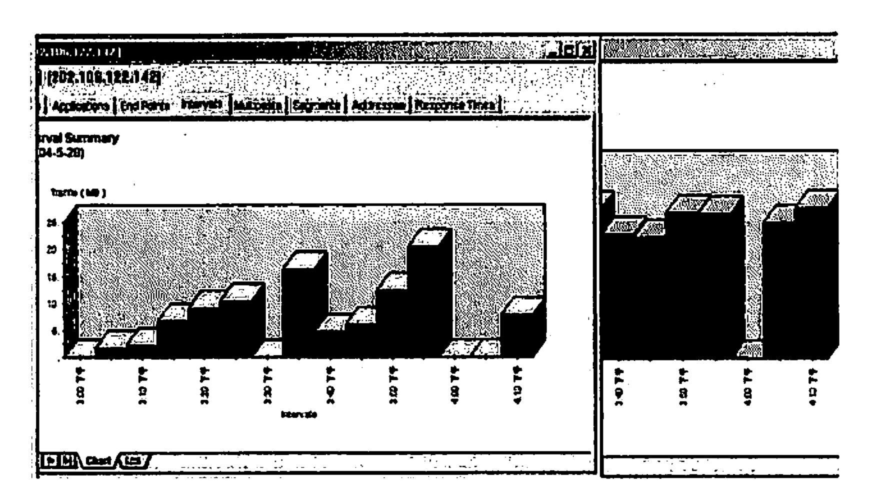
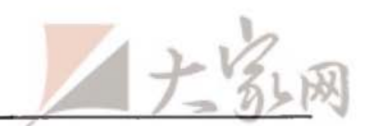


图 20-35　A 工作区流量分布图



- 512Kb 带宽（服务器时间 16:55~17:16），其流量分布如表 20-46 所示，A 工作区流量分布图如图 20-36 所示，机房出口流量分布图如图 20-37 所示。

表 20-46 512K 带宽流量分布

	A	机房出口
应用流量（MB）	57.62	81.32
总流量（MB）	107.87	651.17
百分比	53.4%	12.5%

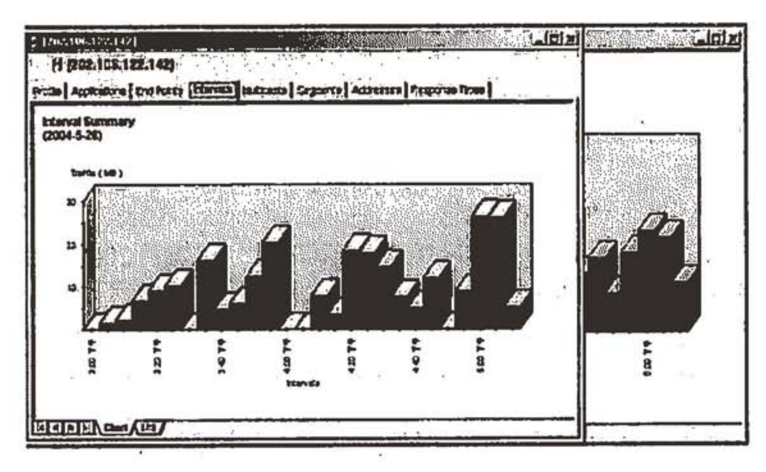


图 20-36 A 工作区流量分布图

- 机房出口流量分布图如图 20-37 所示。

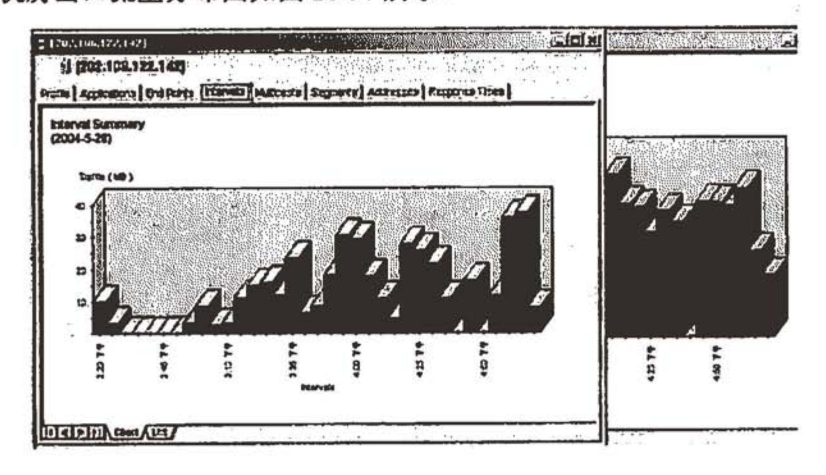


图 20-37 机房出口流量分布图

② 结论：从不同带宽的数据对比来看，随着网络带宽的不断减小，进行同样交易所传输的数据量有较大的增长趋势。这种现象说明随着网络带宽的下降，网络重传现象不断加重，这与前面应用网络故障定位的结论一致。

- 应用分布分析。

① 数据。如图20-38、如图20-39、如图20-40所示，分别是在14:58~15:15、15:41~16:08、16:55~17:16时刻在A工作区进行的不限带宽、2Mb带宽和512Kb带宽的并发压力测试，在机房出口处进行监测得到的，并对机房出口进行了10Mb、2Mb、512Kb的带宽模拟，给出了各种带宽下的带宽利用率。

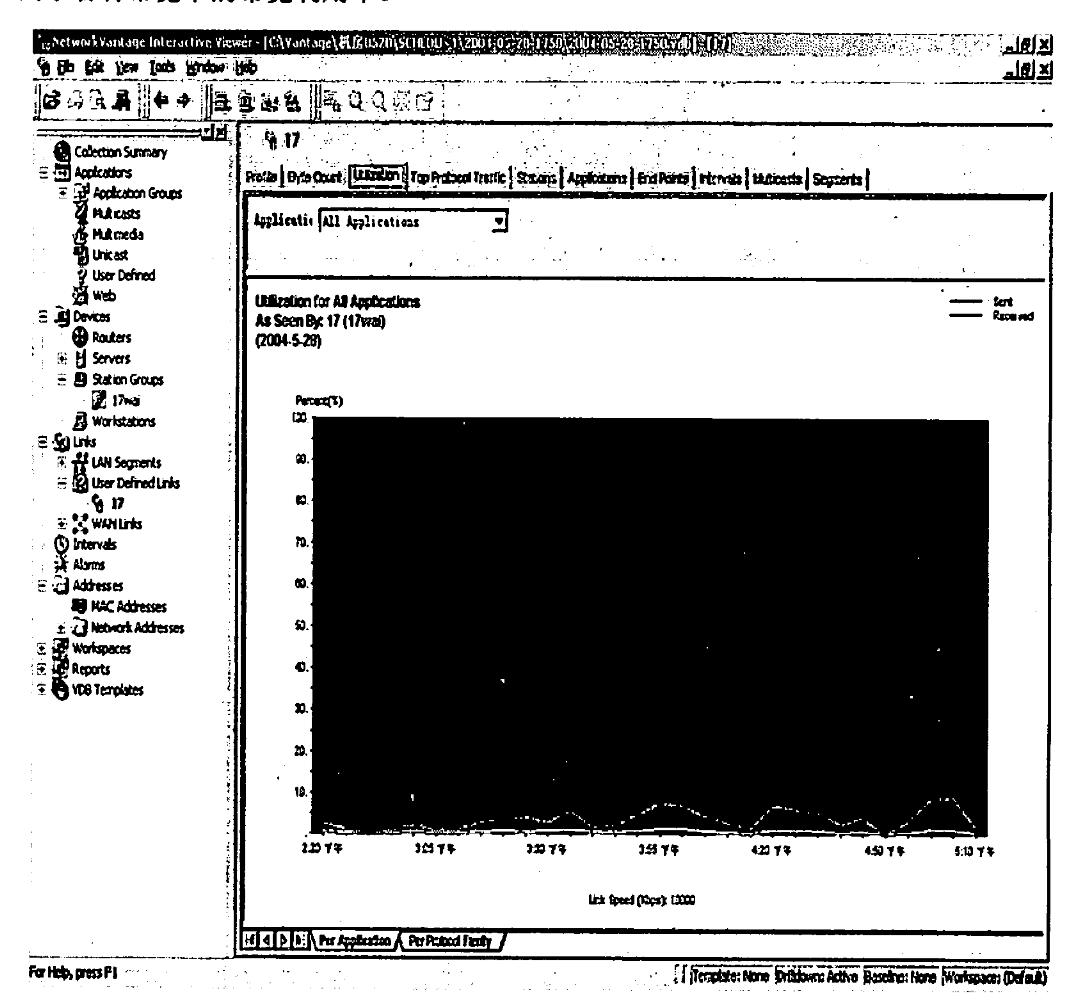


图 20-38　模拟带宽为 10Mb 时的带宽利用率，最大值接近 15%



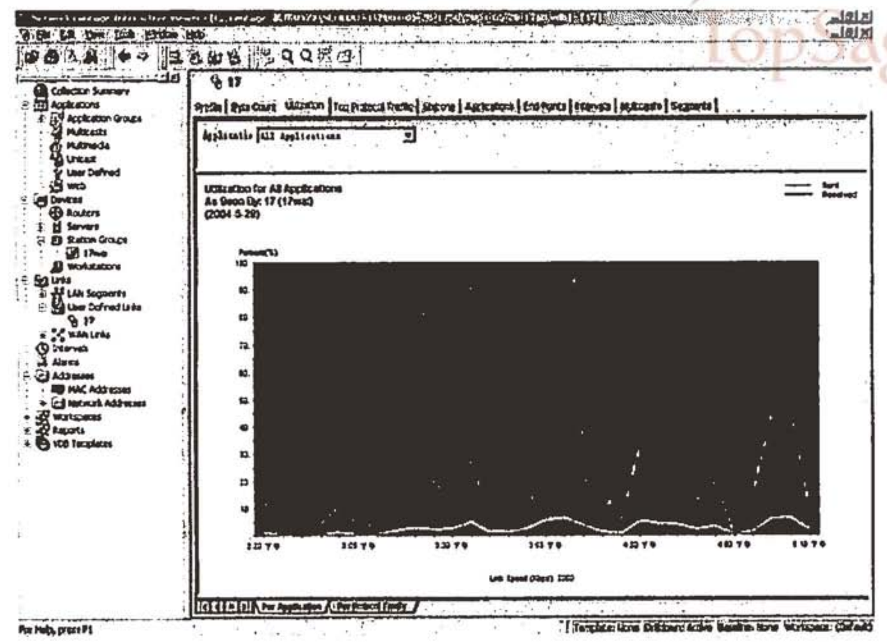


图 20-39　模拟带宽为 2Mb 时的带宽利用率，最大值接近 50%

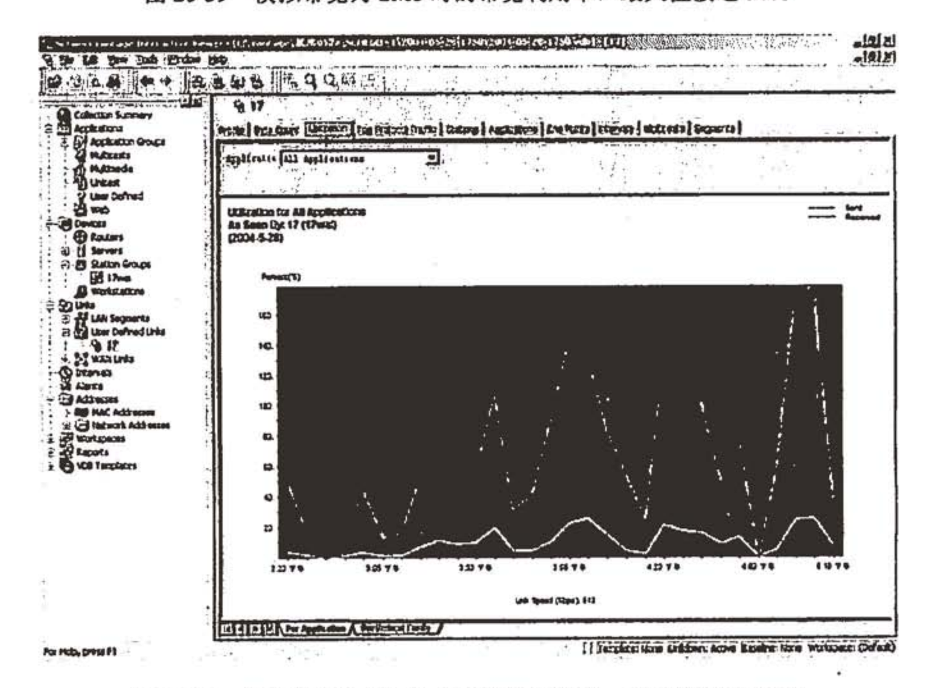


图 20-40　模拟带宽为 512Kb 时的带宽利用率，最大值接近 200%

② 结论：对比不同带宽下的带宽利用率模拟图可以看出，当带宽为10Mb时，带宽利用率最大值接近15%；带宽为2Mb时，带宽利用率最大值接近50%；带宽为512Kb时，带宽利用率最大值接近200%（这里的模拟的带宽为独占带宽）。

（2）防火墙延迟分析。

① 数据。如表20-47、表20-48和表20-49所示的响应时间，分别是在14:58~15:15、15:41~16:08、16:55~17:16时刻，在A工作区进行的不限带宽、2Mb带宽和512Kb带宽的并发压力测试，同时在机房出口和机房服务器端进行监测得到的。

表 20-47　不限带宽响应时间

响应时间 / 探针	最小响应时间（ms）	最大响应时间（s）	平均响应时间（ms）
路由后探针	< 1	35.99	213.49
内网探针	< 1	22.76	199.63
时间差			13.86

表 20-48　2Mb 带宽响应时间

响应时间 / 探针	最小响应时间（ms）	最大响应时间（s）	平均响应时间（ms）
路由后探针	< 1	35.99	214.59
内网探针	< 1	22.76	200.19
时间差			14.40

表 20-49　512Kb 带宽响应时间

响应时间 / 探针	最小响应时间（ms）	最大响应时间（s）	平均响应时间（ms）
路由后探针	< 1	35.99	221.76
内网探针	< 1	34.34	207.39
时间差			14.37

② 结论：对比不同带宽下监测到的防火墙延迟，可以看到延迟值较稳定，基本保持在14ms左右。

通过观察相同时间段两个探针监测到的应用数据，发现数据量相差很小，这说明防火墙的丢包率不是很大，结合前面的分析可以看出，丢包现象主要发生在广域网传输段。



20.8 测试评估与测试报告

20.8.1 局域网测试评估

1. 对比测试环境下的测试结果及故障定位

- 并发用户数未达到性能需求（100）。
- 业务交易响应时间较长。
- Portal 处理速度慢，CM 处理速度快。瓶颈在 Portal，CM 处于空闲状态。
- Portal CPU 资源占用量大。
- 有失败 SQL 调用存在，造成客户端交易失败。

2. 真实业务测试环境下的测试结果及故障定位

- 仍存在对比测试环境出现的所有问题。
- Portal 服务器内存在操作结束后不完全释放，如表 20-50 所示。

表 20-50 Portal 服务器内存统计

操作步骤	Portal服务器内存占用（%）
服务器重新启动	11.3
启动Domino	14.5
启动应用系统	44.3
压力测试结束时刻	81.7
压力测试结束15小时之后	69.5

- 首页 Portal 的 7 个 channel 加载期间 CM 服务器 CPU 资源占用量过大，达到 100%。
- 机房环境下交易的响应时间较长，机房环境与局域网环境对比，基本结论是：在局域网环境下交易的响应时间，比在机房环境下交易的响应时间要长，差值较大的检查点如表 20-51 所示。

表 20-51 机房环境与局域网环境对比差值较大的案例及检查点

案 例	差值较大的检查点
制度文档	无
项目管理	登录、新增项目
工作记事	登录、新增工作记事、进入工作记事、选择日期

- 客户端交易失败时，数据库报错“连接失败”。数据库的最大连接数（目前设

为 40）和 WebSphere 的最大连接数匹配关系，在某种程度上导致该错误。

- 进程级的 CPU 与内存资源占用情况如表 20-52 所示，可以断定进行压力执行期间，Java 进程在争用资源。

表 20-52　进程级的 CPU 与内存资源占用情况

案　例			CPU%（top4）				Memory%（top4）物理内存/虚拟内存			
			1	2	3	4	1	2	3	4
工作记事		进程	Java	Java	Java	httpd	Java	Java	Java	Java
		数值	45	5	3	0.8	14/29	12/24	5/11	1/3
项目管理		进程	Java	Java	Java	server	Java	Java	Java	Java
		数值	80	7	5	0.8	17/26	15/24	5/9	2/3
制度文档	信息上传	进程	Java	Java	Java	Db2agent	Java	Java	Java	Java
		数值	40	5	3	2	19/26	17/24	5/8	1/2
	文件上传与下载	进程	Java	Java	Java	server	Java	Java	Java	Java
		数值	34	30	3	0.8	21/28	18/25	5/8	1/2

- 网络测试环境下必须在承受的并发用户数及交易响应时间之间取得平衡。

3．优化重点考虑

优化重点考虑 Portal 服务器。

- Portal 与 CM 的功能划分，即 EJB 的业务逻辑划分是否合理。
- CM、Portal、WebSphere、DB2 的调优。
- 源代码在开发过程中的优化，例如是否考虑连接缓冲池、动态和静态请求调用是否合理等。
- CM 的调优从应用逻辑、CM 服务器性能两方面入手。
- 将操作之间的关联度降低。
- 数据库的最大连接数以及 WebSphere 的最大连接数匹配关系应该重点考虑。

调优步骤应该为：

- 首页 Portal。
- 响应较慢的网页组件。
- 影响并发用户数的链接。
- 系统硬件资源。

20.8.2　广域网测试评估

1．交易响应时间对比

对比局域网和广域网交易平均响应时间，同一交易在相同带宽下，广域网测试环境



下的响应时间，都比局域网测试环境下的响应时间长，响应时间差如表20-53所示。

表 20-53 局域网与广域网交易响应时间差

案例 / 响应时间差	登录(init_Transaction)	交易1(Action1_Transaction)	交易2(Action1_Transaction)
不限带宽（s）	275.202	216.775	509.822
2M带宽（s）	14.672	62.051	251.035
512K带宽（s）	23.577	20.774	136.185

2．应用网络故障定位

- 广域网环境中网络传输时间所占比例较高，Portal 服务器发出的数据在广域网络上的传输时间，相对服务器本身处理时间要长。网络带宽的影响和其他应用程序争用带宽的影响不是主导因素。因而重点考察 Portal 服务器的网络发送时间。
- 考察 Portal 服务器发出的数据帧的明细，可见应用数据在广域网上存在很多帧丢失和 TCP 重传的现象。（Error：Frame only seen on sending side. prossibly lost by network；TCP Retransmission of earlier frame xxxx）。通过分析数据中的 TCP ACK 帧，可知 TCP 重传，不是由客户端接收缓存满，而不能及时接收数据造成的。要综合考虑网络品质问题和数据发送端的有关 TCP 重传参数设置。
- 例如，针对起始序号 784、结尾序号 973 的数据帧进行包跟踪分析，序号 784 的数据丢失，导致多次重传，序号 791、819、855、973 均为该数据帧的重传帧，且重传的延时不断递增（重传开始时间间隔为 1.407542578sec、3.000778311sec、6.001580091sec）。因此造成应用数据传输时间增加。
- 分析出错重传的数据帧，发现字节少的帧（如 60 字节的 TCP ACK）较少出现丢失重传，字节多（如 1514 字节的 HTTP[HTTP/1.1 200 OK]）丢失重传较多。
- 数据传输的包大则增加了帧的传输次数，同时增加了出错概率。

3．应用网络性能分析

（1）应用程序会话分析

- 针对不同的案例，Portal 服务器到 CM 服务器会话往返行程基本保持一致，可以看到系统应用通信模式基本一致，有利于系统维护。
- 负载生成器到 Portal 服务器运行在广域网环境下，往返行程次数越多，受网络品质、网络延迟影响越大。例如“项目管理”案例达到 138 次。
- 建议降低负载生成器到 Portal 服务器的往返行程次数。

（2）应用程序线程分析（以“项目管理”案例为例）

- 应用往返行程（App. Turns）最大的三个线程受网络品质、网络延迟影响最大。
- 分析往返行程最大的三个线程主要性能数据，其中线程“HTTP:[GET /wps/Portal HTTP/1.1]”与“TCP: 54760<->50000”持续时间最长，并且线程“TCP: 54760<->50000”发出和接收的字节数和帧数最大。
- 线程之间系统间歇（gap）达到 3 秒，可见系统空闲时间在控制范围之内，基本可以接收。
- 线程的请求与回应基本在 1 秒钟之内，但也有约 3 秒钟的响应时间。
- 本案例中不存在相同的请求再次发出的现象。
- 线程与会话（bounce）以及包（packet）之间相关联，包之间的相对时间（relative time）决定了包传输的效率。
- 建议对关键线程进行相关调优，有必要参考包数据。

（3）交易峰值分析（以“项目管理”案例为例）

- 交易最高峰值达到 2.4Mb，1Mb 以下的峰值分布较均匀。
- 交易最高峰值 2.4Mb 所对应的执行线程是调优的关键。
- 建议降低交易最高峰值，调整这一时刻某些线程执行的起始时间。

（4）帧数据分析

- 小于 100 字节和大于 1024 字节的帧在传输的帧中占的比例最高。
- 模拟不同带宽（例如：10Mb、2Mb、512Kb、56Kb）、延迟（例如：150ms、100ms、50ms、1ms）、负载（例如：10%、50%）等网络应用情况，进行应用响应时间预测。

4．网络品质监控

（1）网络流量分析

从不同带宽的数据对比来看，随着网络带宽的不断减小，进行同样交易所传输的数据量有较大的增长趋势。这种现象说明，随着网络带宽的下降，网络重传现象不断加重，这与前面应用网络故障定位的结论一致。

（2）应用分布分析

模拟不同带宽下的带宽利用率，可以看出：当带宽为10Mb时，带宽利用率最大值接近15%；带宽为2Mb时，带宽利用率最大值接近50%；带宽为512Kb时，带宽利用率最大值接近200%（模拟带宽为独占带宽）。

（3）防火墙延迟分析

对比不同带宽下监测到的防火墙延迟，可以看到延迟值较稳定，基本保持在14ms左右。

通过观察相同时间段两个探针监测到的应用数据，发现数据量相差很小，这说明防火墙的丢包率不是很大，结合前面的分析可以看出，丢包现象主要发生在广域网传输段。



附录：测试工具介绍

1. 配置/过程管理工具

（1）TestDirector

开发公司：Mercury Interactive Corporation

工具简介：

TestDirector™是业界第一个基于 Web 的测试管理系统，它可以在公司组织内进行全球范围内测试的协调。通过在一个整体的应用系统中提供并且集成了测试需求管理、测试计划、测试日程控制以及测试执行和错误跟踪等功能，TestDirector 极大地加速了测试过程。

TestDirector 能消除组织机构间、地域间的障碍。它能让测试人员、开放人员或其他的 IT 人员通过一个中央数据仓库，在不同位置就能互通测试信息。TestDirector 将测试过程流水作业——从测试需求管理，到测试计划，测试日程安排，测试执行以至出错后的跟踪——仅在一个基于浏览器的应用中便可完成。

TestDirector 的需求管理可以让测试人员根据应用需求自动生成测试用例。通过提供一个直观机制将需求和测试用例、测试结果和报告的错误联系起来，从而确保完全的测试覆盖率。

TestDirector 的 Test Plan Manager 在测试计划期间，为测试小组提供一个关键要点和 Web 界面来协调团队间的沟通。Test Plan Manager 指导测试人员如何将应用需求转化为具体的测试计划。这种直观的结构能帮助定义如何测试应用程序，从而使组织明确的任务和责任。

一旦测试计划建立后，TestDirector 的测试实验室管理为测试日程制定提供一个基于 Web 的框架。它的 SmartScheduler™根据测试计划中创立的指标对运行着的测试执行监控。

TestDirector 的出错管理直接贯穿作用于测试的全过程，以提供管理系统终端—终端的出错跟踪——从最初的问题发现到修改错误，再到检验修改结果。

（2）AllChange

开发公司：英国 Intasoft 公司

工具简介：

AllChange 将变更管理和配置管理集成一体，从而在应用过程中实现了工具、服务和实施的无缝集成，有效地实现了从概念到发布的全过程。在自动任务模式和连贯生存

周期的支持下，使开发队伍的效率得到提高；信息畅通和准确评估使全体队伍在整个开发周期中保持步调一致；对软件、硬件、文档以及问题递交、变更和发布过程的集成管理，使企业可以开发出满足客户需求的高质量产品。

AllChange 主要功能包括：变更管理、配置项管理、发布管理、过程管理和文档管理。其强大的基线功能可以定义不同版本各自的配置项、自动锁定各项配置并拒绝任何改动，从而排除了发布内容的不确定性。它还支持不同变更过程的集成，包括变更配置项、控制和发布变更等内容，最大程度的自动和透明的过程使得过程管理变得更为有效。

AllChange 是一个开放的系统，不仅可以与 Telelogic Doors、I-Logix Rhapsody、Artisan Real Time Studio、Visual Studio、Visual Basic、Visual Age for Java 等开发环境和平台集成在一起，还可以通过 API 与 Office Tools，IDE，Help Desks 和 E-mail 等其他系统之间实现双向集成，可直接在这些产品内部进行 check in 和 check out 操作。

AllChange 通过定义不同的生存周期，便捷地实现了对项目的有效管理。除单机操作模式外，AllChange 还支持 Client/Server 模式和 Browser/Server 模式，其灵活的多用户模式和强大的可伸缩性，可以满足各种用户的不同需求。

（3）IRqA

开发公司：TCP S.I.公司

工具简介：

IRqA 是一个提供对于整个需求工程过程支持的集成的需求管理工具，通过 IRqA 用户可以创建高质量的系统需求描述，捕获和管理需求，分析需求关系。IRqA 具有以下特点：提供基于需求工程过程的简便易用的、强大的集成环境；Filtering system 提供方便的需求捕获、管理和修订功能；Blocks 和 Domains 提供有效的需求组织和分类；IRqA 提供自动的需求关系分析、描述和测试；支持报告和文档的自动生成；Access partitions 提供用户访问控制和管理；提供需求描述的可跟踪性；可以通过 XMI 接口，支持项目或者子项目与其他工具，如 UML 工具间的导入和导出；支持 ODBC 接口，提供开放的数据信息存储。

（4）Telelogic DOORS

开发公司：Telelogic 公司

工具简介：

Telelogic DOORS 是一款需求管理工具。它可以捕获、连接、跟踪、分析和管理信息，以便于确保实施的工程与需求规格说明和标准相一致。

DOORS 可以提供企业团队工作中的通信能力、协从能力和需求验证能力。其直观的用户界面可以方便地帮助多用户通过网络并发地访问，并且能够维护大量的管理对象（需求和关联信息）和连接。fish-eye 和 Microsoft Windows 资源管理器图形方式的管理

视图，使每一个用户都可以方便地定制想要看到的需求信息。

DOORS包括一套完整的变更建议流程和审核系统，使用户可以对需求递交变更建议。内部的项目连接允许项目共享需求、设计和测试，以及提高需求的跟踪能力。讨论机制支持用户针对一个意见协从合作交流，以加快意见或想法的确立、执行、转换和实现。

Distributed Data Management（DDM）分布数据管理支持远程用户临时访问和使用DOORS的所有功能。然后再离线工作，并且远程用户可以将数据更新到主数据库中——这使异地团队成员和子承包商可以方便地合作开发和通信。

此外，DOORS 为用户提供了无限制关系的、多级的、用户可自定制的跟踪能力。它的跟踪向导可以跨多级地生成连接报告，并且在相同的视图中显示——提供 fool-proof 周期确认和验证。

DOORS提供了电子表格风格的面向文档数据视图，与Microsoft Word和Excel有很好的集成。同时具有强大的可伸缩的管理能力，支持多平台操作，具有基于网络针对多用户并行工作的方式提供管理大型复杂项目的能力。

（5）CM Synergy

开发公司：Telelogic 公司

工具简介：

CM Synergy通过支持基于任务的配置管理，使团队的所有成员的开发活动进行有效地协同与沟通。CM Synergy的基于任务的方式，提供了通常影响多个文件的逻辑变更（如修改与增强功能）的流程驱动；从一个基线及其附加的与一系列任务相关联的变化所产生的配置；通过一个"安全与智能"的配置管理系统可以发现编译的软件是否完成且与变更保持一致；通过图形界面驱动来定制Telelogic的模板流程，可以很容易适应团队所熟悉的流程，使之符合用户习惯的开发方法；通过对用户展示开发项目的真实状态的综合视图来加强对项目的管理。

CM Synergy内部的分布式变更管理（Distributed Change Management），可以使分布在不同地域的开发团队共享信息。DCM能使开发者在不同地点共享与管理共同的软件部件和变更需求。由于CM Synergy DCM不仅能够把项目数据和变更需求发布到远程的团队，也能够发布开发流程，这使得它成为一种分布式环境。流程模板可以在一个地方被定义与集中管理，然后再发布到远程的团队，保证各地的团队在重要的控制方面保持一致且容易维护。

CM Synergy可以与WebSphere Studio、Eclipse、Visual Studio、Visual C++、Visual Basic等开发环境相集成。此外，通过引入ActiveCM.ActiveCM，为用户提供了自动且透明的CM解决方案。它为开发者提供了新的界面与使用模式，使与其开发环境透明地集成。

（6）ChangeSynergy

开发公司：Telelogic 公司

工具简介：

Telelogic ChangeSynergy™为基于网络的变更需求管理建立标准，是完全基于网络的变更需求跟踪和报告系统。与Telelogic的变更管理解决方案CM Synergy 完全集成，ChangeSynergy简化了变更需求管理的进程，使企业可以从外部和内部资源对变更作出反馈。通过为团队和企业提供可靠的、基于任务的端到端及进程驱动的自动操作，实现了对整个开发生命周期的控制。通过ChangeSynergy，用户可以跟踪、管理各种各样的软件和信息资产，基于网络的应用软件及站点，满足不同用户的需求，同时执行组织和业务的需求。

ChangeSynergy 为整个企业的开发环境提供了基于任务的变更管理平台，用来管理变更需求。通过与 Telelogic 的变更和配置管理库集成，使企业可以使用基于任务的方法来管理以软件为中心和以变更为中心的需求。该系统使企业可以跟踪和记录所有的变更需求，并且将所有开发活动的管理都可视化。它还将管理与项目状态及他们需要的其他信息相连，从而作出迅速的决定，保持进程的持续性。

ChangeSynergy 提供了人性化的直观的变更需求界面，使任何管理或团队成员都可以轻易地创建变更需求以及在整个可用的模板、过程和进程中分配和跟踪任务。ChangeSynergy 系统基于角色并且可以提供容易使用的任务分配、监控和报告性能。

ChangeSynergy 提供现成的模板、流程和过程，结合 Telelogic 基于任务的配置管理工具，一个端到端的变更管理系统能被很快地使用起来，从而解决开发组织所面临的效率和质量问题。

（7）McCabe TrueChange

开发公司：McCabe&Associates，Inc.

工具简介：

McCabe Truechange 是第三代软件配置管理工具，提供直观的、逻辑性的方法以管理软件的版本及更改。由于是基于任务方式监控软件的更改情况，Truechange 简化了软件配置管理过程，具有监控管理不同版本或不同平台下软件变化或改变的能力，同时可无缝隙地支持开发过程，并有利于同一软件多个应用版本的维护。

Truechange 是基于 Change-Sets 技术的软件配置从单一的文件更改上升到逻辑功能更改，如软件错误的修正、软件功能的更改。

Change-Sets 技术允许用户移动、共享和熟练操纵各种变化，并把它们看作一个独立的系统。具有支持任何更改时间表、安全性与可追踪性、支持交叉平台、与 QA 及测试产品集成等特点。

（8）T-Plan

开发公司：T-Plan Ltd.

工具简介：

T-Plan 是一款测试过程管理工具，支持任何规模的项目从单元测试直到用户验收测试的各项测试过程，无论是手工测试还是自动测试，技术人员和非技术人员都可以计划、编写文档，管理软件测试活动、工程的商业目的实现和质量保证。

T-Plan 的核心是清晰的管理过程。它来自预置的过程结构。这种过程可以局部修正或者替代公司的标准过程。测试过程管理基于业界广泛采用的“V”模型。通过使用 T-Plan，测试计划将作为测试过程的一部分结合在一起。其主要功能如下。

① 测试分析：这是测试过程管理的核心。定义被测系统如何集成和评审；定义功能结构图和每个功能的相关风险因素等。

② 测试执行管理：在这个阶段，测试统计、覆盖率和测试成功与否等测试结果被记录下来。在任何阶段管理层可以确定什么测试已经执行过并且通过了，什么测试已经执行过并且失败了，什么测试还没有执行，什么测试还没有完成不能运行。

③ 事件管理：所有的事件可以记录下来，分类和跟踪直至解决。

④ 能力：T-Plan Professional 是所有测试资产的中心存储数据库。各个资产之间相互联系并且随着测试的进行捕捉数据，既提供有价值的资源用来管理当前测试工程，并且帮助用户计划未来的工程。

⑤ 报告和分析：T-Plan Professional 提供测试过程的广泛的报告，用户可以定义报告中需要什么。

⑥ 可追溯性：通过按照过程驱动的方法测试，可以在软件生命周期中管理变更。从需求而来的测试条件和测试脚本之间的关系可以提供这些信息，从而提高了可追溯性。

⑦ 与其他工具的连接：T-Plan Professional 与需求管理、测试自动化（捕捉/回放）和其他软件开发工具可以连接，这种连接是由软件和最小用户交互来控制的。

2. 功能测试工具

（1）WinRunner

开发公司：Mercury Interactive Corporation

工具简介：

WinRunner™是一种企业级的用于检验应用程序是否如期运行的功能性测试工具。通过自动捕获、检测和重复用户交互的操作，WinRunner 能够辨认缺陷并且确保那些跨越多个应用程序和数据库的业务流程在初次发布时就能避免出现故障，并且保持长期可靠运行。

用 WinRunner 创立一个测试，您只须记录下一个标准的业务流程，WinRunner 直观的记录流程能让任何人在 GUI 上单击鼠标就可建立测试。

在记录一个测试的过程中，可插入检查点，在查询潜在错误的同时，比较预想和实际的测试结果。在插入检查点后，WinRunner 会收集一套性能指标，在测试运行时对其一一验证。

除了创立并运行测试，WinRunner 还能验证数据库的数值，从而确保交易的准确性。

WinRunner 的 DataDriver™ Wizard，单击鼠标就能简单地将一个记录下的业务流程转化为一个数据驱动的测试，来反映多个用户各自独特且真实的操作行为。WinRunner 的另一个能加强测试质量的特征是具有 FunctionGenerator™——一种可视工具，能快速可靠地为您的测试增加功能。使用 FunctionGenerator，可以从目录列表里选择一个功能加到测试中以提高测试能力。

WinRunner 执行测试时，它会自动操作应用程序，它的意外处理功能为测试排除干扰，包括消息和警报。一旦测试运行后，WinRunner 的互动式的报告工具通过提供详尽的、易读的报告，其中会列出在测试中发现的差错和出错的位置，来帮助您解释所得的结果。这些报告对在测试运行中发生的重要事件进行描述，如出错内容和检查点等。

（2）SilkTest™

开发公司：Segue Software，Inc.

工具简介：

SilkTest 是用于对企业级应用进行功能测试的产品，可用于测试 Web、Java 或是传统的 C/S 结构。SilkTest™ 提供了许多功能，使用户能够高效率地进行软件自动化测试。这些功能包括：测试的计划和管理；直接的数据库访问及校验；灵活、强大的 4Test 脚本语言，内置的恢复系统（Recovery System)；以及具有使用同一套脚本进行跨平台、跨浏览器和技术进行测试的能力。

① 跨浏览器和跨平台测试：SilkTest 通过使用 Document Object Model (DOM) 技术，实现了对 Netscape Navigator 6.0 的支持。DOM 技术确保测试脚本在不同平台、不同浏览器下具有可移植性，以提高测试效率。同一个测试脚本可以不需要修改就运行于 Windows NT、Windows 2000、Windows 95/98、Windows Me、Windows XP、Netscape Navigator 4.0/6.0、Microsoft Internet Explorer 4.0/5.0/5.5/ 6.0。

② 使用 4Test 语言进行强力测试：SilkTest 的 4Test 是面向对象的第四代语言，特别适用于复杂测试。SilkTest 中的所有测试案例，无论是录制生成或是脚本生成，均使用 4Test 语言创建。4Test 语言提供了大量的命令、数据类型和函数，允许用户通过循环和分支结构扩展测试用例的范围，也可以在测试用例中包含例外处理，从而保证脚本的强壮性。

③ 分布式应用的集中测试：SilkTest 的分布式测试结构，可以同时跨越 Windows 和 UNIX 前端、浏览器以及基于 Java 的网络系统环境运行同一个测试。

④ 对质量过程的有效管理：SilkTest 提供了跨多平台、多开发环境、多浏览器的测试计划、开发、执行及结果报告。通过 SilkTest，可以共享测试计划、语汇、测试案例等。

此外，SilkTest 能够与其他 Silk 产品有机的结合在一起，例如面向过程测试管理的 SilkPlan Pro，它包括需求的检查及确认，测试执行的安排和产品是否具备发布条件的评估报告功能。也可以与面向工作流缺陷管理的 SilkRadar 进行集成。

3. 性能测试工具（系统强度测试工具）

（1）LoadRunner

开发公司：Mercury Interactive Corporation

工具简介：

LoadRunner®是一种预测系统行为和性能的负载测试工具。通过模拟成千上万名用户和实施实时性能监测来确认和查找问题，LoadRunner 能够对整个企业架构进行测试。通过使用 LoadRunner，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。

LoadRunner 是一种较高规模适应性的自动负载测试工具，它能预测系统行为，优化性能。LoadRunner 强调的是整个企业的系统，它通过模拟实际用户的操作行为和实行实时性能监测，来帮助您更快地确认和查找问题。

使用 LoadRunner 的 Virtual User Generator 引擎，能很简便地创立起系统负载。用 Virtual User Generator 建立测试脚本后，可以开始对其进行参数化操作，这一操作能让您利用几套不同的实际发生数据来测试应用程序，从而反应出本企业的工作负载。

LoadRunner 内含集成的实时监测器，在负载测试过程的任何时候，都可以观察到应用系统的运行性能。这些被动监测器为您实时显示交易性能数据，用户可以在测试过程中从客户和服务器的双方面评估这些系统组件的运行性能，从而更快地发现问题。

一旦测试完毕后，LoadRunner 收集汇总所有的测试数据，并提供高级分析和汇报能力，以便迅速查找到性能问题并追溯原由。

（2）SilkPerformer V™

开发公司：Segue Software，Inc.

工具简介：

SilkPerformer V™ 是一种在工业领域最高级的企业级负载测试工具。它可以模仿成千上万的用户在多协议和多计算的环境下工作。不管企业电子商务应用的规模大小及复杂性，通过 SilkPerformer V™ 均可以在部署前预测它的性能。可视的用户化界面、实时的性能监控和强大的管理报告可以帮助用户迅速地解决问题。

SilkPerformer V™可以使用并且管理位于远端的测试负载代理机器，并可以顺利通过防火墙。它支持互联网安全标准，包括身份验证，使 SilkPerformer V™ 能够测试电子

商务系统所有方面的任务。一个简单强大的，基于 Pascal 的脚本语言提供了对所有用户活动的控制。

在独立的负载测试中，SilkPerformer V™ 允许用户在多协议、多计算环境下工作。并可以精确地模拟浏览器与 Web 应用的交互作用。它的 TrueCache™ 技术为模拟浏览器的 Cache（缓存）提供了精确的保证。TrueModem™ 技术和工作负载模块使 SilkPerformer V™可以产生可信的负载。对客户端的 IP 地址与 DNS 的模拟允许用户测试负载平衡。

SilkPerformer V™中的 TrueLog™ 技术提供了完全可视化的原因分析技术。通过这种技术，可以对测试过程中的用户产生和接收的数据进行可视化处理，包括全部嵌入的对象和协议头信息，从而进行可视化分析，甚至在应用出现错误时都可以进行问题定位与分析。

通过附加的 Server Analysis Module（Server 分析模块 SAM），可以监测服务器的统计数据，并与负载测试结果数据结合起来，识别系统中后端服务器的问题，甚至包括防火墙后端的服务器。

同时，利用 SilkPerformer V™对中间件与数据库的支持，可以对系统中的后端服务器进行负载测试，并修改任何系统伸缩性和性能问题。

此外，用于提高可扩展性的开放性工业标准接口，确保了 SilkPerformer V™也能够用于那些 out-of-the-box 功能并不完善的特定场合。在最新的 SilkPerformer V™中还提供了对 Microsoft .NET 的支持；TrueScale™ Web 回放器提供广泛的页面校验和错误记录功能；可进行基于场景测试的 All-Day 负载测试等新特性。

（3）Benchmark factory

开发公司：Quest Software，Inc.

工具简介：

Benchmark Factory 是一种高扩展性的强化测试、容量规划和性能优化工具。它通过模拟成千上万个用户访问应用系统中的数据库、文件、Internet 及消息服务器的真实情形，方便而准确地确定充裕的系统容量，查找系统瓶颈，隔离出分布式计算环境中与系统强度有关的问题。

Benchmark Factory 通过记录和回放多用户测试中的事务处理过程，使 IT 专业人员能够更加方便地完成系统强度测试。用户进程、实景延迟和 Benchmark Factory 的强大脚本引擎，强化了测试的彻底性和简便性。

除了模仿交易量外，Benchmark Factory 还可通过对输入时间、思考时间和执行间隔的设置，模拟出交易发生前，用户输入数据，用户在进入下一步操作前作出决定的过程之类的活动，从而更加真实地模拟交易场景。

此外 Benchmark Factory 为 IT 人员提供了不同的测试结果分析方法。用户可以对多个测试进行比较，或进一步挖掘某个测试的细节。除了分析功能外，还可以打印更全面的报表，使用户能够通过便于阅读的图形和表格掌握大量的信息。

（4）JProbe

开发公司：Quest Software，Inc.

工具简介：

JProbe 是一款 J2EE 性能测试和诊断工具。JProbe 套件中包含了四个模块：JProbe Profiler、JProbe Memory debugger、JProbe Threadalyzer 和 JProbe Coverage，分别从 Java 代码性能、内存使用状况、线程状态和测试代码覆盖量四个方面对 Servlet、JSP 和 EJB 应用代码进行诊断、分析。

JProbe Profiler 模块内置了 Call Graph（调用关系图）和高级数据采集机制，可以将性能问题落实到方法和代码行的层次。主要功能包括：确定方法的热点，以逐行的方式挖掘分析性能状态，计算运行时间或 CPU 占用时间，确定要诊断分析的关键代码区域，预测代码修改对性能造成的影响。

JProbe Memory Debugger模块可帮助开发人员快速查找Java代码的内存泄露和对象循环现象。内置的实时内存使用情况图和对象视图有助于开发人员掌握实际的内存使用状况，并设法减少内存消耗以提高应用性能。主要功能包括：通过易用的两步分析法，跟踪运行时的内存增长状况，计算内存泄露量，通过Leak Doctor发现可能的内存泄露根源，检测过多的短期对象和垃圾收集详情，确定代码改变对内存使用的影响。

JProbe Threadalyzer模块是一种强大的线程问题探测工具，可以发现诸如死锁、停顿、竞争等危及应用数据完整性的问题。主要功能包括：观察线程状态的变化，预测死锁及数据竞争状况，发现不当同步情况避免数据损坏。

JProbe Coverage模块可以帮助开发人员查找测试的代码量，精确计算已测试的代码量，轻易实现测试工作可靠性和精确度的评估。主要功能包括：迅速确定未测试代码或死代码；分析特定条件下所覆盖的代码范围；精确的覆盖范围报告；以XML、纯文本、CSV或者HTML格式输出覆盖范围报告，实现个性化分析。

4．白盒、嵌入式测试工具

（1）TAU Logiscope

开发公司：Telelogic 公司

工具简介：

Logiscope 是一组嵌入式软件测试工具集。它遵循 ISO/IEC9126 定义的“质量特征”标准，对软件的分析，采用基于国际间使用的度量方法（Halstead、McCabe 等）的质量模型，以及从多家公司收集的编程规则集，从软件的编程规则、静态特征和动态测试覆

盖等多个方面，量化地定义质量模型，并检查、评估软件质量。

在开发阶段，用户不仅可以从 RuleChecker 预定义的 50 个的编程中选择规则，而且也可以用 Tcl、脚本和编程语言定义新的规则。此外，还提供了 50 个面向安全-关键系统的编程规则，帮助用户建立规则集和定义质量模型。RuleChecker 用所选的规则对源代码进行验证，指出所有不符合编程规则的代码，并提出改进源代码的解释和建议，并可通过文本编辑器直接访问源代码，指出需要纠正的位置。

Audit 以 ISO 9126 模型作为质量评价模型的基础，将被评价的软件与规定的质量模型进行比较，以图形方式显示软件质量的级别，并对度量元素和质量模型不一致的地方作出解释并提出纠正的方法。

在测试阶段，Logiscope 推荐对指令（IB）、逻辑路径（DDP）和调用路径（PPP）的覆盖测试。此外，对安全-关键软件还提供了 MC/DC 的覆盖测试。TestChecker 产生每个测试的测试覆盖信息和累计信息。并根据测试运行情况实时在线更改，随时显示新的测试所反映的测试覆盖情况，从而确保测试的有效性。

在维护阶段，Logiscope Audit 将应用系统的框架以文件形式（部件文件间的关系）和调用图的形式（函数和过程间的关系）可视化，从而减少对未知系统理解所需的时间。

Logiscope 在代码开发和评审阶段，可寻找潜在的错误。通过对未被测试代码的定位，Logiscope 帮助找到隐藏在未测试代码中的缺陷。其为开发者提供 SEI/CMM 在第 2 级（Repeatable）所要求的软件质量跟踪等关键实践的要求，推进开发组织尽快达到 SEI/SMM 的 3 级。

（2）McCabe IQ

代理公司：McCabe&Associates，Inc.

工具简介：

McCabe IQ 是一款静态分析 / 软件质量度量工具集，包括 McCabe EQ、McCabe Test、McCabe Reengineering 等组件。它提供客观的方法评估测试覆盖率，具有一致性、重复性、客观解释、质量评估、优化测试、分配资源，管理具有危险性的软件等功能。能真正保证测试每个独立的路径，可确定测试过的路径和没测试过的路径。

McCabe EQ 集成一个新的 SQL 服务器，用于收集来自于 McCabe IQ 诸多工程的静态和动态数据。通过将这些数据信息上载至 EQ，可以在一个用户自定义的层次上去管理数据信息，整合软件工程结果。对于生成的测试报告，用户可以自行修改。

McCabe Test 以 NIST（国家标准技术局）出版的测试标准为基础，能彻底地对系统进行测试，找出系统中的错误。通过自动化、标准化测试过程，可以缩短测试周期，对测试完整性进行审查，指导更有效的测试，精确地进行测试资源计划分配。McCabe Test 的白盒测试技术能够更多地覆盖被测软件，找到更多错误、从而提高被测软件可靠性。

McCabe 还可根据部分用户更高端的测试需求，提供再工程软件模块 McCabe Reengineer。

此外，McCabe IQ 可以对给定的软件记录多达 105 种度量；提供了 35 种预定义的报告供用户选择，这些报告从最简单的表格到图形化数据表示可以帮助本地（模块级）和系统分析。为提高基本代码分析结果的综合性，报告有超链接能力，允许用户从相关数据关联到系统组件；McCabe IQ 还提供了 battlemap 和 flowgraph 可视化工具，具有各个级别上的可视化图形显示能力。

（3）PRQA C/C++

代理公司：Programming Research 公司

工具简介：

PRQA C/C++是代码规则静态检测工具。它可为用户提供针对 C 或 C++代码的静态检测、复杂度分析以及程序分析。该工具的最大特点在于，可以根据用户的需求，进行代码规则的专门定制，满足不同行业用户的不同需求。

另外，针对 C 语言的 QA C 工具还可以与目前流行的 MIRA 标准进行无逢集成，提供更多、更规范的代码规则。

（4）C Verifier（C 程序验证器）

代理公司：法国 PolySpace 公司

工具简介：

PolySpace C Verifier 是用来直接进行运行错误和非确定性构件检测的工业化产品，在编译时针对 ANSI C 的应用进行。PolySpace Verifier 不需要代码执行和修改，或运行耗时的测试用例。PolySpace Verifier 能精确地指出引起运行错误的代码部分。

PolySpace C Verifier 依靠的是建立在抽象解释上的静态验证技术。已有的测试和调试工具是动态和交互地验证软件的一致性，抽象解释技术使得 PolySpace Verifier 能够彻底地发现和检查出那些在未来的运行中出错的代码。

PolySpace C Verifier 具有 Static Verification 和 Abstract Interpretation 功能，静态验证(Static Verification)检查软件应用的动态特性，并没有实际的运行，节约了大量的时间。抽象解释（Abstract Interpretation），是一种提供工业规模的源代码的有效分析，仅使用较小的计算机资源。通过使用抽象技术，PolySpace Verifier 对被分析的软件和派生的动态特性进行抽象，而不是交互地验证软件状态。PolySpace Verifier 中的切边技术提供非常高效的选择性（诊断出安全、不安全和费代码）并且极大地缩短彻底的代码评审的时间要求。

PolySpace Verifier 由基本内核和配置与启动分析的图形化用户接口组成，可自动和彻底地检查下面的错误：企图读未初始的变量、多线程应用中未保护数据的访问冲突、

对空指针和越界指针的引用、对超界数组的访问、非法类型转换（long to short, float to integer)、非法的算数运算（如除零错误，负数开方)、整数和浮点数的上溢出/下溢出、不可达到的代码。

（5）ASG-SmartTest

开发公司：Segue Software，Inc.

工具简介：

ASG-SmartTest 是 ASG-ESW（Existing System Workbench）上的交互式测试器和调试器，ASG-SmartTest 提供自动的测试和调试功能，帮助用户发现各种语言和平台上的错误和结构设计问题。

ASG-SmartTest 提供 TSO、Batch、Dialog Manager、CICS 和 IMS 域上运行的 COBOL、PL/I 和汇编语言的程序的源码级调试功能，并对支持的所有语言和平台提供统一的界面接口，用户可以快速地建立、执行和记录应用程序测试过程。

ASG-SmartTest 可以使用户快速地定位程序中需要测试的问题，执行路径和数据元素，提高应用程序质量；可以让用户在测试过程中动态调整数据值和内存，以减少反复测试的需要。此外，ASG-SmartTest 用户可以确定程序中所有对应的部分和条件都已经被测试过，标示出那些还需要进一步测试的部分，增加用户对被测试代码的信心。

（6）Cantata++（汉化版）

开发公司：Information Processing，Ltd.

工具简介：

作为一款单元 / 集成测试工具，Cantata++提供了动态测试、代码覆盖率分析、静态分析等功能，可以验证被测软件是否满足设计要求；检测被测软件在进行动态测试时代码执行覆盖率，用于分析动态测试效果及指导软件测试；分析被测软件源程序复杂性及软件结构，用于分析被测软件可维护性及分析被测软件是否符合编程标准。

Cantata++ 4.0 版本新增加了 Windows 平台的 Borland C++、Builder 6.0；Symbain 平台的 UIQ 和 Series 60 编译器。在即将发布的新版 Studio 中将增加自动生成中文测试报告的功能，测试脚本向导的用户界面有进一步的提高和增强，用户可以在测试脚本向导中创建一个完整的 C/C++的测试驱动程序，可以在界面中使用或关闭单个测试用例，并且增加了对 Studio 的数据输出的选项。

Cantata++4.0 可完全集成任意的嵌入式开发环境，和常用开发环境 MSVC++6.0，BorlandC++Builder 的 IDE 界面完全集成，对于命令行的开发环境，例如 Linux 平台的 ERC32 开发环境 CCS2.0.6/RTEMS/SIS Similator，Cantata++可在 makefile 中和编译环境完全集成。

（7）UniTester

代理公司：奥索公司（参与开发）

工具简介：

UniTester 是一款汇编语言单元测试工具。它可以在不增添任何硬件仿真器或数据发生器的前提下，高效率、低成本地完成对 8031/51、8086/88、386、8096/98、TMSC3x、TMSC6x、1750A 汇编语言软件的单元测试任务。其主要功能如下。

① 被测模块和测试用例的管理与维护：UniTester 提供一整套由驱动模块、桩模块和 I/O 描述文件共同组成的测试用例的维护过程。包括编辑测试用例、配置测试用例、维护测试用例（测试用例与被测模块的一致性检查功能，包括被测模块状态的监测、显示和处理）。

② 提供每个被测单元（模块）动态测试覆盖率指标：每个测试用例经动态测试后，统计出的语句覆盖率和分支覆盖率；以及在进行多次测试后，对语句覆盖率和分支覆盖率的累计统计。

③ 提供每个被测单元动态测试时的性能指标：详细地列出每次动态测试过程中，被测程序的执行过程（包括分支的走向、语句执行的次数、语句占用的时间、跳转的方向以及次数等），用户可以通过这些信息，完全了解程序的执行过程，发现和分析问题。

④ 对设定要追踪的对象（包括程序或数据内存、片内内存、寄存器、以及变量），在测试结束时，给出其测试过程中的变化过程（包括最大值、最小值、均值以及每次数据变化的过程）。

⑤ 对设定要进行数据比对的对象（包括程序或数据内存、片内内存、寄存器、以及变量），在每次动态测试结束后，给出其比较结果，并自动写入测试报告。

⑥ 在每次动态结束后，自动给出当前所有 CPU 寄存器的状态值。

⑦ 完善的嵌入式硬件环境模拟机制：UniTester 提供一套完善的嵌入式硬件环境模拟机制，其目的是任何嵌入式软件可以不经任何修改，即可在模拟环境中运行，并完成单元测试工作。包括完善的 I/O 描述机制和完善的中断描述机制。

5. 软件开发工具

（1）Telelogic TAU

开发公司：Telelogic 公司

工具简介：

Telelogic TAU Generation 2（简称：TAU G2）是基于 UML 2.0 语言的系统工程和软件工程开发环境。它是一个可视化的系统工程工具集，包括系统设计与分析工具 TAU Architect、系统实现工具 TAU Developer、系统测试工具 TAU Tester。TAU G2 是基于角色的，为不同目标用户群提供不同层次的功能。同时，这些工具又是建立在同一平台上的，提供了统一的用户界面，从而缩短了学习周期，加速了系统的实施。

Telelogic TAU/Architect 是一个用于复杂系统分析和描述的现代的、基于模型的系统工程工具。它支持可视化语言 UML 2.0，对大型复杂系统全面建模，从而得出详细、易于理解并且是明确的描述。使用 TAU/Architect 后，系统工程师不仅能描述结构，还能在开发中描述系统的行为。描述还可以被仿真来验证系统，并可以在开发早期向最终用户与其他投资人展示系统的行为。

Telelogic TAU/Developer完全具备TAU/Architect的功能，并能够生成高效C、C++或Java代码，其中Agile C代码专门面向嵌入式系统。

Telelogic TAU/Tester是基于通用测试语言TTCN-3，用于自动和集成测试的强大工具。TTCN-3是一个现代的且灵活的语言，通过广泛的接口用于描述许多类型的系统测试。典型的应用领域为系统测试、交互性测试、协议测试、服务测试、模块测试等。TTCN的平台独立性和其特殊的测试能力，使得它被广泛地应用于定义通迅系统的正式测试集。

除TAU第二代工具外，TAU还包括了基于UM1.4，对复杂应用软件进行分析、建模及面向对象设计的工具TAU UML Suite；协议一致性测试工具环境TAU TTCN Suite；软件质量保障和度量工具TAU LOGISCOPE；基于SDL和MSC的协议开发工具TAU SDL Suite。

（2）Real-time Studio

开发公司：英国 ARTiSAN 公司

工具简介：

Real-time Studio 是一套适合于多用户进行项目开发的开发工具，包括 Real-time Studio Professional 和 Real-time Modeler。应用 Real-time Studio 提供的一系列开发工具，项目开发小组可以实现软件及系统设计的安全可靠、自动有效的移植。

Real-time Studio Professional 能够协调项目小组各个成员的工作，使项目的立项、设计及文档管理更好地满足实时系统的要求，同时使用户的工作具有较高的可维护性。Real-time Studio Professional 对基于 UML 的建模方法在实时系统设计方面进行了大量的扩展。

Real-time Studio Professional 提供了对逻辑顺序图的动画显示，对状态模型的模拟，同时还集成了 Altia 的图板显示，应用 Professional 提供的这些工具用户，可以对系统及软件行为的有效性进行充分的验证。

Professional 高效的同步工具可以使项目工程师实现 C、C++、Java 源码与系统模型框架的双向转换，保证源码设计与模型设计的一致。

此外，Real-time Studio Professional 还集成了 RTM Workshop 及 Telelogic DOORS，来实现在整个项目周期中对需求管理的支持。开发人员可以在 DOORS 或 RTM 中对实时系统的项目需求进行设计，在 Real-time Studio Professional 中完成系统的定义、设计，

将不同的需求设计与项目设计各个阶段中的不同模型链接起来。

在配置管理方面，Real-time Studio Professional提供了两种方案与配置管理（CM）工具及源码控制工具进行链接。一种是应用微软的CSCC标准为各种模型及包文件提供直接的版本支持，另一种是通过Real-time Studio Professional OLE实现与配置管理的自动链接。

6．其他

（1）SilkTest International™（测试平台）

开发公司：Segue Software，Inc.

工具简介：

SilkTest International™是适用于当今全球企业级应用的一种先进的，基于标准的测试平台。通过 SilkTest International，Segue 扩充了其功能测试产品 SilkTest® 的能力，使用户通过执行单一测试脚本，同时测试跨多语种、平台和 Web 浏览器的应用。该系统具有以下特点。

① 利用单一脚本测试多种语言：通过 SilkTest International，开发商可以更好地安排产品本地化的时间进度。一个单一的测试脚本可以支持所有语种，各种语种的测试也可以同时进行，加速了产品面市时间，降低了软件测试费用。

② 符合 Unicode 标准支持双字节：SilkTest International 为任何语种的应用测试提供完整的基于标准(Unicode)的支持。对于双字节字符的全面支持可以使其在以前的标准 ASCII 码所不能支持的语言环境（如日文，简体中文等）下进行测试。

③ 对本地化平台的广泛支持：SilkTest International 确保经过本地化的应用能够正确地运行在当地的软件和硬件环境中。一个测试脚本，不需更改，就可以同时在多个 Windows NT 和 Windows 2000 本地版下运行。由于对多种开发平台的支持，测试开发和执行时间更进一步的减少，这些平台包括 HTML、JavaScript、ActiveX、Java、Visual Basic 和 C/C++。SilkTest International 同样承认国际通用的键盘，为与本地化密切相关的数据提供全面的处理，例如日期和数字，确保和本地版操作的一致性。

④ 有效的质量管理过程：SilkTest International 无缝集成了跨多平台、多开发环境、多浏览器的计划、测试和报表功能。使用 SilkTest International，用户可以共享测试计划、语汇和执行成组的基于用户定义标准的测试用例，这些都是集中控制的。此外，自动化的特点通过 SilkTest International 独特的恢复系统得以进一步增强。如果发现某个错误并且使应用发生中断，恢复系统会记录这个错误并使应用重新复位到原状态，以便下一个测试用例能够运行。

（2）Bender-RBT（需求分析 / 测试用例生成工具）

开发公司：美国 TBI 集团

工具简介：

Bender-RBT 是基于需求的测试用例设计工具，可完全自动地设计功能测试用例。因为 Bender-RBT 是基于功能需求的，所以可用来测试不同语言、不同平台的应用系统。在使用时，只需产生文本文件以定义被测功能模块，Bender-RBT 将自动决定所要求的功能变量，并通过功能变量生成测试用例。用户不需具备逻辑知识，也不必了解算法如何使用因果图工作，所有这些都由 Bender-RBT 自动完成，易于使用。

在功能方面，Bender-RBT 以可读英文格式产生测试用例，这使得非技术人员也可以评审测试；该工具可评估现成的测试用例库，并可设计补充需要的测试用例以达到 100%的功能覆盖，这对维护管理测试用例库非常有用。作为自动退化测试工具的前端工具，Bender-RBT 可与许多捕捉/回放工具接口集成使用。此外，Bender-RBT 提供测试用例追踪性需求，它可产生符合 MIL-STD-498（前身为 MIL-STD-2167A）的部分文档。

通过应用 Bender-RBT，可以达到以下效果。

① 增加测试的有效性：Bender-RBT 通过设计测试用例可达到 100%的功能覆盖。代码覆盖率超过 70%～90%，接近两倍于典型测试的 30%～50%代码覆盖率。

② 增加测试效率：因为 Bender-RBT 采用精确的数学方法（明感路径算法）设计测试用例，所以测试用例没有冗余，这样，退化测试库比通常的方法要小 30%～50%。

③ 缩短开发周期：Bender-RBT 可与产品分析、设计及编程各阶段同时进行，早期的错误检测可减少项目费用并可节省时间。许多 Bender-RBT 用户在编程之前即已进行功能测试。

（3）ADS-2（系统测试平台）

开发公司：北京旋极信息技术有限公司

工具简介：

ADS-2 是以通用的工业标准为基础的分布式的实时系统。它的 I/O 和实时子系统运行在 VME 主机上，VME 主机提供仿真运行的 CPU 资源。用户接口系统对通用的 UNIX 平台也是可用的，并提供所有需要的工具以配置、控制实时系统，实时系统发射、控制仿真的运行。监控、分析数据流并记录数据。所有功能都同时对用户有效。

通过使用接口控制文档（ICD），可支持所有的数据分析和数据定义，为了系统设计，ICD 提供了相关的数据库，允许所有的信息和数据格式定义到参数级。不需要用户过多地介入，描述性语言可以进行重复性测试、运行脚本。

作为开放系统的 ADS-2，提供通用第三方软件接口，如 Matrix X 仿真代码发生器和 VAPS。Workbench 或 TCL/TK 可显示系统原型和运行情况。函数库提供了与任意的模拟应用程序和虚拟仪器程序，如 LabView 和 HP VEE 的接口。模拟和接口原型能够通过 TCP/IP、共享 RAM 或高带宽的确定网络，如 FastLink 或 ScramNet，访问 ADS-2。



作为模块化的系统，ADS-2 可不同规模地满足特殊环境下的特别需要，如 I/O 类型，I/O 资源的数量，处理能力和功能（分析、数据采集、模拟）的要求。ADS-2 系统可以配置成在典型的实验室环境和非常恶劣的环境中使用，如飞行测试环境。

ADS-2 提供的主要功能包括数据采集与分析、模拟、仿真与激励、测试支持函数（TSF）程序、TECL 用户扩展、VME 实时计算和 I/O、FIBO 故障注入与获取单元、仿真、激励和模拟等。

（4）Datafactory（数据库装载工具）

开发公司：Quest 公司

工具简介：

Datafactory 是一款能够在短时间内将大量数据装载到数据库的工具软件，只要几次鼠标单击就能够根据具体要求，生成一个含有数百万行数据的数据库。其主要功能包括：所生成的数据具有测试的实际意义，例如姓名、地址、年龄和邮政编码都是模拟实际情况生成的；数据量足够大，能够模仿用户访问高峰时的状态；可以和 Oracle、DB2、Sybase、SQL Server 等数据库通过 ODBC 直接连接；允许用户使用自己的数据信息（比如文本形式的数据）对数据库进行装载。

（5）EasyLinux Development Kit for ARM

开发公司：Micetek International Inc.

工具简介：

EasyLinux Development Kit for ARM 是针对 ARM 学习者和开发工程师提供一个完整的解决方案。这一方案包括 Hitool 集成开发环境、GNU 编译器、Linux 开发包、TCP/IP 网络协议栈、文件管理系统、图形包、Embedded JTAG 在线仿真器、ARM 开发板、板级驱动软件包（BSP）和参考设计原理图。

其集成开发环境 Hitool 使基于 Windows 平台下，Linux Kernel 裁减-编辑-Kernel 编译调试-应用程序编译调试统一于同一开发环境，并可实现 Kernel 跟踪功能和 Kernel aware debugger。

7. 仪器仪表

（1）安捷伦 N3900A 模块化光网络测试仪

开发公司：安捷伦公司

工具简介：

安捷伦 N3900A 模块化光网络测试仪为安装、运行和维护光网络提供了一个重量轻、坚固耐用的便携式测试系统。采用模块化设计，可在您需要时提供各种所需的测试功能。模块化测试平台使 N3900A 具有多种测试方案组合，具有强大的可扩展性。

它集 OTDR 功能、损耗测试功能、DWDM 功能及 PMD/CD 色散测试功能于一身。

具有触摸屏与按键操作方式，以及安捷伦专利的浏览键。屏幕图形显示的内置光显微镜用于快速检查光接头。N3900A 具有“业务检测”功能，可以防止用户在工作光纤上错误地启动 OTDR 测量。测试采用最精确的琼斯矩阵算法。具有大容量固态存储器，有效防震动和撞击。

（2）FrameScope350 局域网性能分析仪

工具简介：

FrameScope350 局域网性能分析仪是能够使用客观的性能指标自动测量，并报告常见网络业务性能的手持式分析仪。可方便、自动、迅速的帮助用户识别网络运行状态，包括计算机节点、协议统计分布及网络响应时间等。同时具有线缆认证功能，可以快速测试 6 类及 E 级线缆。

FrameScope350 局域网性能分析仪可以测量和确定网络中服务器、文件服务器、电子邮件服务器、打印 DNS 和 DHCP 等关键资源的响应时间。Ping、TraceRoute 和 SNMP 查询功能可以测试到用户定义的设备的连接能力。流量发生功能可以测试网络在重载情况下的性能。MAC 环回可以测量以太网中的吞吐量、损耗、时延和偏差。全方位网络测试，能够对协议、配置、性能和布线故障迅速定位和排除。通过网络发现功能，用户可以检查网络，创建包括工作站、交换站、路由器、服务器和其他网络设备的完整列表。采用简便易用的彩色触摸屏界面，可以生成电缆图形测试报告，并可实现远程控制。

（3）LANpilot 掌上型网络分析仪

工具简介：

LANpilot 掌上型网络分析仪是一款掌上型网络协议分析解决方案。它提供了实时网络统计、流量分析，自动识别各种协议和协议解码。只需简单地将 LANpilot 连接到网络中，就能方便地进行网络维护和故障定位。

LANpilot 掌上型网络分析仪具有强大的网络统计功能，不仅可以按照网络节点、IP、VoIP、数据包、字节数、包长、广播数进行统计，更可以按照业务应用（FTP、Telnet、Mail、Web、News、VoIP、NetBIOS、SNMP）进行统计，轻松统计网络利用率、错误率，自动生成带宽占用矩阵图、流量统计表。

LANpilot 掌上型网络分析仪可对协议进行详细解码、分析，可实时进行性能监测，解码 2、3 层，TCP/IP 和应用协议。其强大的过滤功能，帮助用户自由选择关注的数据。此外它具有 Ping、TraceRoute 功能、自动节点发现、吞吐量测试、PPPoE 测试、WLAN 测试、报告中心、在线帮助以及无线打印等功能。同时支持 IPv6、10/100M 以太接口，接入方便。

（4）SuperLink 网络测试仪

工具简介：



SuperLink 网络测试仪作为多网络、多规程测试和分析的综合平台，面向固定网和移动网的网络建设、运营维护、设备调试和网络优化分析，已在电话交换网、智能网、GSM/CDMA/GPRS 移动网、移动智能网、ISDN 网和接入网中得到了广泛应用。

SuperLink 网络测试仪一表多用，能满足电信维护的全面需要，支持的协议包括 MTP、TUP、ISUP、SCCP、TCAP、INAP、GSM-MAP、GSM-CAP、GSM-A 接口、GSM-Abis 接口、GB 接口、CDMA-MAP、CDMA-A 接口、V5.1/V5.2、DSS1/PRA、1 号随路信令等。

SuperLink 网络测试仪具有在线监测、协议解析、协议统计、呼叫/事务分析和网络特性分析功能、强大的数据库分析功能、自定义过滤、数据捕捉、专家分析功能，帮助用户迅速定位故障，深入剖析问题原因。具有多种数据输出方式，如数据文件、text 文本、Excel、Bmp、数据库、自定义报表或直接打印。

SuperLink 网络测试仪单机可配置 16 条双向 2Mb/s 链路，每条链路支持全时隙（31/32）、N×64Kb/s 或高速 2Mb 信令方式，单机可测链路高达 256 条。并可基于多前端联网或数字交叉连接系统（DXC）链路收敛测试，配置多达 496 条双向 2Mb/s 链路，可用于建立大型测试系统或监测系统。

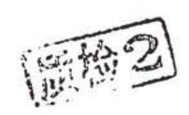
（5）WireScope350 线缆认证测试仪

工具简介：

WireScope350 线缆认证测试仪可用于光缆、电缆日常维护测试和“综合布线系统”质量认证测试。测试精度超过 TIA、ISO III 级标准，并可跟踪实验室标准。

WireScope350 线缆认证测试仪测试项目涵盖了线序图、电缆长度、近端串扰、衰减、等效远端串扰、回波损耗、环绕噪声、传播时延、环路电阻等国际、国内最新标准的要求。可认证安装的局域网布线系统是否符合国际、国家标准，包括 TIA 三类、五类、超五类、六类和 ISO D 级和 E 级标准。单端可测线缆长度，对线缆中的每对线准确故障定位。识别错接、短路、开路、反接和线对分开，检测屏蔽层的连续性。

WireScope350 线缆认证测试仪单端即可测试线缆长度，对线缆中的每对线进行准确故障定位。可选单、多模光纤测试模块，额外附赠光源、光功率计测试功能，一机多用，性价比极高。LED 指示灯直观方便地显示测试进度、测试失败原因等。支持对讲功能，通用 CF 存储卡，USB 数据传输接口。附带的“ScopeData Pro”软件可生成专业质量的图形测试报告，自动生成、打印标签，并提供完善的在线帮助。



全国计算机技术与软件专业技术资格（水平）考试指定用书

根据人事部、信息产业部文件，计算机技术与软件专业技术资格（水平）考试纳入全国专业技术人员职业资格证书制度的统一规划。通过考试获得证书的人员，表明其已具备从事相应专业岗位工作的水平和能力，用人单位可根据工作需要从获得证书的人员中择优聘任相应专业技术职务（技术员、助理工程师、工程师、高级工程师）。计算机技术与软件专业实施全国统一考试后，不再进行相应专业技术职务任职资格的评审工作。


ISBN 7-302-10536-7
9 787302 105367 >
定价：60.00元